

APPLICATION OF A CONSTITUTIVE MODEL TO EXTENSIONAL AND SHEAR
RHEOLOGY OF POLYSTYRENE CARBON NANOFIBER COMPOSITES

Undergraduate Research Thesis

Presented in Partial Fulfillment of the Requirements for Graduation with Honors Research
Distinction in the College of Engineering of The Ohio State University

By

Nathan W. Volchko

William G. Lowery Department of Chemical and Biomolecular Engineering

2015

Thesis Committee:

Dr. Kurt W. Koelling, Advisor

Dr. Lisa Hall

Copyright by

Nathan W. Volchko

2015

Abstract

This research focuses on improving an existing constitutive model to predict the rheology of polystyrene-carbon nanofiber (CNF) composites and examining the differences between two nanocomposite preparation techniques and two types of nanofibers. The constitutive model is a set of equations adapted to predict the dynamic behavior of polymer nanocomposites and the orientation evolution of nanofibers. The equations in the model are functions of time, type of flow, stress, deformation, nanofiber orientation, and flow history due to the polymer's viscoelasticity. Important parameters in the model include: polymer relaxation time, polymer viscosity, mobility factor, polymer-particle interaction, particle-particle interaction, and aspect ratio. Three flow fields were examined: small-amplitude oscillatory shear, transient shear, and transient extensional flows. The two preparation techniques examined are commonly used in preparing polymer composites: melt-blending and solvent casting. The nanofibers examined were as received from the manufacturer with some undergoing additional high-heat treating. This research resulted in the development of an accurate model for the composites for all three flow fields. Values of parameters in the model have given insight into the physical behavior of the composite due to polymer-particle and particle-particle interactions and due to agglomeration. An accurate model of a polymer-CNF composite would be beneficial to industry, as the manufacturing processes such as extrusion, flow through dies, or spraying alter the orientation and consequently the mechanical, electrical, thermal, and optical properties of the composite. This model can provide a way to understand these effects would improve the optimization of properties.

Acknowledgements

I would like to express my deep appreciation for my advisor, Dr. Koelling, for all of his assistance throughout this project. His encouragement and guidance made this research a remarkable experience and has provided me with motivation for advanced study. I will truly miss continuing researching with him. I would like to thank Timothy Kremer for his assistance in helping me start this project and providing insight into his programs. I would also like to thank Christopher Kagarise on whose work this research stems from. Finally, I would like to thank Koki Miyazono for all the experimental data analyzed in this research.

Vita

2011.....Woodridge High School
2015.....S. Chemical Engineering, Ohio State University

Publications

Murch, William; Kremer, Timothy; Janko, Joseph; Brooks, Lukas; Volchko, Nathan;
Koelling, Kurt. (in press) Rheology of polystyrene/carbon nanofiber composites
during flow reversal experiments. J. Rheol. Acta.

Field of Study

Major Field: Chemical Engineering

Minor Field: Mathematics

Table of Contents

Contents

Abstract	ii
Acknowledgements	iii
Table of Contents	v
List of Figures and Tables.....	vii
Chapter 1: Introduction.....	1
Chapter 2: Materials and Methodology	7
Chapter 3: Description of Constitutive Model	11
Chapter 4: Optimizing the Model.....	17
Parameter fitting for pure polymer	17
Parameter fitting for composite	20
Divergence	25
Chapter 5: Results and Discussion	29
Pure polymer	29
Composite	34
Chapter 6: Conclusions and Future Work.....	46
Conclusions	46
Future Work	47
References	48

Appendix A: Programs	55
Constitutive Model Solver: Extensional Flow	56
Constitutive Model Solver: Shear Flow	59
Pure Polymer: Overhead Parameter Optimization	62
Pure Polymer: SAOS Flow	65
Pure Polymer: Transient Shear Flow	67
Pure Polymer: Transient Extensional Flow	72
Determining Boundary of Convergence for σ , λ , and α	75
Composite Parameter Optimization: C_1	82
Composite Parameter Optimization: Aspect Ratio Scaling Factor	84
Composite Parameter Optimization: σ	91
Composite Model Predictions: Melt Blended with O-CNFS (MB)	94
Composite Model Predictions: Melt Blended with HHT-CNFs (MBHHT)	103
Composite Model Predictions: Solvent Cast with O-CNFs (SC)	111
Composite Model Predictions: Fitting G' and G''	119
Composite Model Predictions: Solving the Constitutive Model for SAOS Flow	121
Composite Model Predictions: Modeling the Stress Wave in SAOS Flow	125

List of Figures and Tables

Figure 1: Single-walled nanotube conformations (a) zig-zag (b) armchair [3]	1
Figure 2: Stacked cup structure of CNFs (angled view) [6]	2
Figure 3: Aligning of CNFs under shear and extensional flow (a) before deformation (random alignment) (b) undergoing shear flow (c) undergoing extensional flow (highly aligned) [14]	4
Figure 4: Distribution of CNF length [41]	9
Figure 5: Coordinate System for Orientation of Single Nanofiber [10]	14
Figure 6: Effect of σ in (a) shear and (b) extensional flow; other parameters fixed: $CI = 0.005, r = 40, 2wt\%$	22
Figure 7: Effect of <i>reflective</i> in (a) shear and (b) extensional flow; other parameters fixed: $CI = 0.005, \sigma = 0.75, 2wt\%$	23
Figure 8: Convergence area for λ and α values at $\sigma = 0.6, 5wt\%$ CNFs, and $\gamma = 1s - 1$ [40]	26
Figure 9: New convergence area for λ and α values at $\sigma = 0.5, 2wt\%$ CNFs, and $\varepsilon = 1s - 1, \gamma = 3s - 1$	27
Figure 10: Divergence Surface for (a) extensional flow (b) shear flow	28
Figure 11: Pure polymer predictions for storage and loss moduli	31
Figure 12: Pure polymer prediction for shear viscosity	32
Figure 13: Pure polymer prediction for extensional viscosity	33
Figure 14: Shape Factor Comparison (a) shear flow (b) extensional flow	35
Figure 15: Moduli Predictions for: (a) MB 2wt%, (b) MB-HHT 2wt%, (c) SC 2wt%	39

Figure 16: Shear Viscosity Predictions for: (a) MB 2wt%, (b) MB-HHT 2wt%, (c) SC 2wt%	41
Figure 17: Extensional Viscosity Predictions for: (a) MB 2wt%, (b) MB-HHT 2wt%, (c) SC 2wt%	43
Figure 18: Orientation Evolution of a_{11} , $\varepsilon = 0.1s - 1$	44
Table 1: Analysis of CNF length after preparation of PS/CNF composites [41].....	9
Table 2: Shape Factors for Various Concentration Regimes	13
Table 3: Optimized Parameter Values for Pure Polymer	29
Table 4: Optimized Parameter Values for Composites	36

Chapter 1: Introduction

Particles are classified as nanoparticles if at least one dimension is in the nanometer scale. Common examples are single-walled (SWNT) and multi-walled carbon nanotubes (MWNT), carbon nanofibers (CNFs), and nanoclays. Interest in nanoparticles has been sparked in industry because nanoparticles are capable of increasing the mechanical strength of composites at lower loadings than conventional macroscopic particles such as carbon black. Thermal and electrical properties of poor conducting media such as polymers may also be improved with the addition of nanoparticles [1]. Such materials are called nanocomposites, combining at least two distinct materials [2].

Single-walled carbon nanotubes are formed of graphite sheets rolled into a tube with a diameter of approximately 1 nm, displaying either zig-zag or armchair conformations [3]. These conformations can be seen below in Figure 1: **Single-walled nanotube conformations (a) zig-zag (b) armchair**Figure 1.

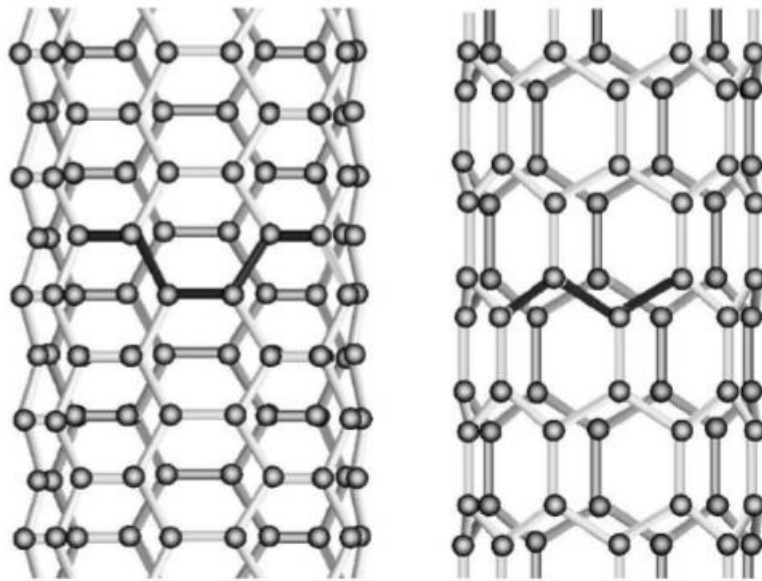


Figure 1: Single-walled nanotube conformations (a) zig-zag (b) armchair [3]

Multi-walled nanotubes are made of concentric single-walled nanotubes with diameters around 10-20 nm. Both of these carbon nanotubes (CNTs) can be 10-100 times stronger than steel at a lower weight [4]. Electrons are able to move along the carbon lattice giving them their good electrical properties [5]. However, the current limitation of their use is their high cost.

CNFs are also tube-like structures, but are much larger than CNTs with diameters of 50-200nm and lengths into the micron range. The structure of nanofibers is also different, displaying graphitized stacked cups structure as can be seen in Figure 2.

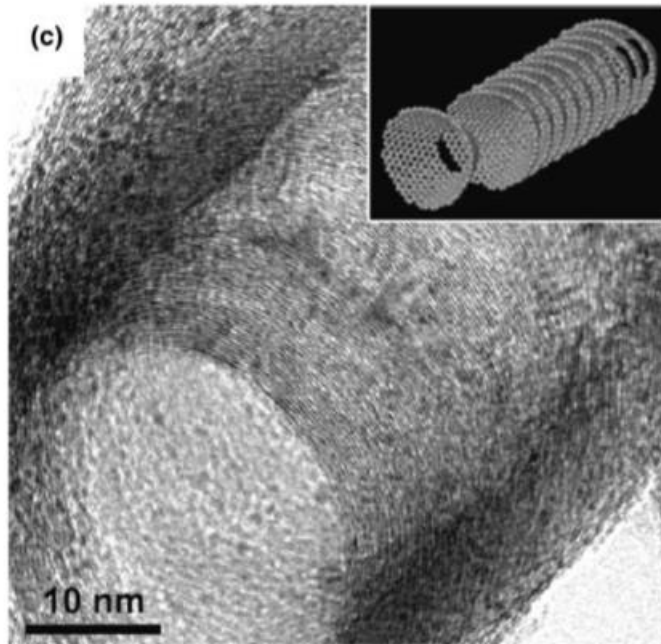


Figure 2: Stacked cup structure of CNFs (angled view) [6]

This structure is not as ordered as CNTs so the mechanical properties of CNTs are better than CNFs; however, CNFs still have very good mechanical properties. Additionally, the edges of

CNFs have improved physical bonding with other materials such as polymers [1]. The advantage to CNFs over CNTs is their availability at 1/500 of the cost [8]. Therefore, since they also display many of the properties of nanotubes, they have a strong potential for use in industry.

One of the common ways of making CNTs and CNFs is chemical vapor decomposition. Organic vapors are heated to high temperatures and in the presence of metal catalysts, the vapors decompose into SWNTs, MWNTs, and CNFs, [7]. These particles can then be further treated to purify them. Methods include high heat treatment and washing with an acid. Particularly with CNFs, both techniques can shorten fiber length and affect how the CNFs interact with each other and with a polymer [1].

One problem for the use of CNTs and CNFs in composites is achieving a high level of dispersion. SWNT tend to clump into ropes, and MWNT and CNFs are often tangled [4]. This is of concern when the nanoparticles are placed in a matrix such as a polymer, where dispersion is desired for optimal mechanical properties [5]. The three common techniques used to improve dispersion with a polymer are melt blending, solution processing, and *in-situ* polymerization [2]. Melt blending involves melting a polymer, and then mechanically mixing in the nanoparticles and allowing the diffusing of polymer chains between the nanoparticles [11]. One negative effect of melt blending when using CNFs is a breakdown of CNF length due to the mechanical mixing; however, melt-blending generally achieves good dispersion of nanoparticles. In solvent casting, a polymer is dissolved in a solvent, the nanoparticles are mixed in by sonication, and then the solvent is evaporated off, forming the composite [12]. Solvent casting preserves the original length of CNFs which is longer than in melt-blending but may not disperse the particles

as well. The final method, *in-situ* polymerization, involves sonicating the nanoparticles in a monomer solution which is then polymerized to give the composite [13].

Another important detail is the dependence of the properties of nanocomposites with the orientation of the CNTs or CNFs. Mechanical and electrical properties are strong along the length of the CNTs and CNFs but are weak in the perpendicular direction [14]. Therefore, properties can be optimized in a particular direction by orienting the particles in that direction. One way to orient the particles in a matrix, such as a polymer, is to flow the melted composite. The degree of alignment is different under different flow fields as can be seen in Figure 3.

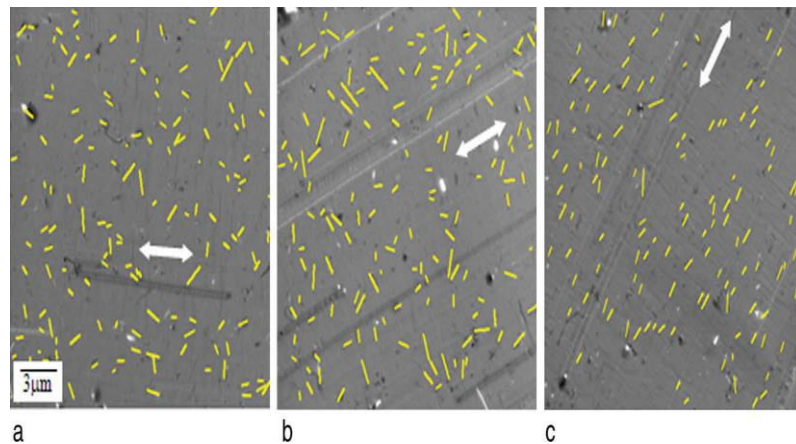


Figure 3: Aligning of CNFs under shear and extensional flow (a) before deformation (random alignment) (b) undergoing shear flow (c) undergoing extensional flow (highly aligned) [14]

Additionally, industrial processes involving nanocomposites could also affect the orientation of the nanoparticles, thus affecting the properties of the final product. Extrusion, injection molding, spraying, and fiber spinning are examples of such

processes. To be able to understand these effects, the rheology of polymer nanocomposites is of interest.

There has been a great interest in researching polymer-CNT composites, with far less research devoted to polymer-CNF composites [15-19]. The research that has described the rheology of polymer-CNF composites for various polymer systems such as polyethylene [20], polypropylene [21-29], polycarbonate [21, 30-32], polyester [33], polyamide [34, 35], poly(methyl methacrylate) [36, 37], and polystyrene [38]. Additionally, these focus only on shear rheology, excluding extensional rheology and the evolution of fiber orientation [14].

Previous research in Koelling's group has addressed some of these deficiencies by researching both shear and extensional rheology of CNFs in polystyrene. A constitutive model for both flow fields was developed by Kagarise [10]. This model was then applied to the shear rheology during forward and reverse flows by Murch and Kremer [39, 40].

This research is focused on improving the accuracy of the model, incorporating shear, extensional, and small-amplitude oscillatory shear experiments. Within the constitutive model is a parameter that can couple the stress of the polymer to the stress due to CNFs. Kagarise did not use this coupling parameter in his studies of shear and extensional flows [10], and neither did Murch in his shear flow reversal experiments [39]. Kremer did allow the coupling effect in his flow reversal and small-amplitude oscillatory shear experiments and found an improvement in the accuracy of the model [40]. One of the improvements in this research is now applying that coupling also to extensional flows, with the goal of optimizing one set of parameters for the constitutive model to describe shear, extensional, and small-amplitude oscillatory shear flows. Two

preparation methods, melt-blending and solvent casting described above, were examined and modeled to characterize differences in the rheology between the two methods.

Rheological differences between two types of CNFs, one as received from the manufacturer and one high heat treated for purification, were also examined.

Chapter 2: Materials and Methodology

A polymer – CNF composite system was examined for all the experiments which had been previously conducted in this lab [41]. The polymer used was polystyrene (PS) as received from the Chevron Phillips Chemical Company LP (MC3600, specific gravity: 1.03, MFI: 13.0 g/ 10 min at 200°C), chosen for its well-characterized rheology, lack of crystallinity, and affinity for CNFs due to the aromatic group in the repeated polymer unit [14].

The two types of CNFs used were as received in powder form from Applied Science with the first being ordinary CNFs (Pyrograf III, type PR-24-PS) denoted O-CNF, and the second being high heat treated to 3000°C (Pyrograf III, type PR-24-XT-HHT) denoted HHT-CNF. O-CNFs have been pyrolytically stripped, meaning polyaromatic hydrocarbons have been removed from the surface of the fibers, whereas the high heat treating in HHT-CNFs graphitizes carbon on the surface, creating the most graphitic CNF offered by Applied Science and having reduced concentrations of iron catalyst in the CNFs. Since the high heat treating improves conductivity, HHT-CNFs would be useful for electronic and thermal applications, and O-CNFs for mechanical and electrical. The chemical vapor deposition process produces entangled fibers in both instances [42].

Two preparation methods for the samples were examined: melt-blending (MB), and solvent casting (SC). MB samples were prepared by adding PS pellets and CNF powder to a DACA twin screw microcompounder, heating to 200°C, and mixing for 5 minutes at 250 rpm. These conditions allowed good dispersion of CNFs without

degrading the PS. Samples were prepared at 0wt% and 2wt%. Once mixed, the composite was extruded through a 2mm die and cut into 2-3mm length pellets [14].

SC samples were prepared by dissolving 19.6g PS in 200 mL tetrahydrofuran (THF), stirring for 12h, and adding CNFs. CNFs were added in 0wt% and 2wt%. The composite was ice-cooled while being sonicated at 20 kHz for 15 min using a Sonic Dismembrator (Fisher Scientific, Model: 550; probe: 1”) at a power level of 550 WL⁻¹. The composite was then film cast at room temperature, dried in a vacuum oven at 65°C, broken by an Analytical Mill (IKA) into 0.5-1mm diameter chips, and dried in a vacuum oven at 65°C for an additional 5 days to remove all THF [14].

Test samples for both preparation methods were created by compression molding of the pellets/chips into 25mm diameter and 0.9-1.2mm thick discs for shear flow tests and 52mm x 7mm x 1.55mm rectangular bars for extensional flow tests. The compression molding was done by melting the pellets/chips at 200°C for 15 min, quickly pressurizing and releasing pressure four times to remove air bubbles, pressurizing for an additional 10 min, turning off the pressure and heat, and removing samples once cooled. Storage of test samples in a vacuum oven for 24 hours at 65°C before rheological measurements was done to prevent absorption of air or moisture [14].

The nanometer-scale diameters of CNFs were measured using a FEI Technai G2 Spirit transmission electron microscope (TEM) at 100keV and 4800 x magnification, and the micron-scale lengths were measured with a Zeiss Axioskop optical microscope at 400 x magnification. The lengths of CNFs after test sample preparation were examined to characterize differences in aspect ratio between the different preparation methods. Figure 4 shows the distribution of lengths. Table 1 shows the average diameters, number

average lengths, weight average lengths, aspect ratios, and polydispersity of average lengths [41].

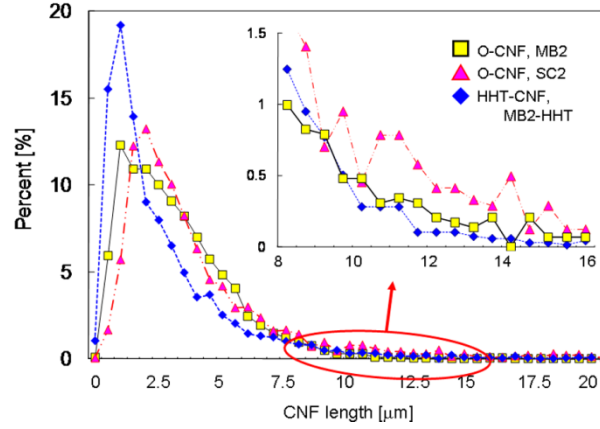


Figure 4: Distribution of CNF length [41]

Table 1: Analysis of CNF length after preparation of PS/CNF composites [41]

Specimens	Diameter [μm]	L_N [μm] ^{*1}	L_W [μm] ^{*2}	% of long CNFs ^{*3}	Aspect ratio ^{*4}	Polydispersity
O-CNF, MB	0.065 \pm 17	3.43	4.78	1.56	53	1.39
HHT-CNF, MB	0.066 \pm 18	2.92	4.64	3.06	44	1.59
O-CNF, SC	0.065 \pm 17	4.46	8.01	7.86	69	1.80

*1: Number average length of CNFs

*2: Weight average length of CNFs

*3: Total percentage of CNFs longer than 10 μm

*4: L_N /Diameter of CNF

*5: L_W/L_N

Of note, the CNFs in the SC composites have the longest lengths and aspect ratios. CNFs in MB and MB-HHT are more comparable, with CNFs in MB being longer than MB-HHT. L_N , number average length, was used in the calculation of all aspect ratios.

Linear viscoelastic behavior was characterized with SAOS. A strain-controlled rheometer, ARES LS2 from TA Instruments, with a torque transducer (0.02-2000 g cm) and a normal force transducer (2-2000 g) was used. Parallel plate geometry with 25mm diameter and 0.9-1.2mm gap distance was used. Molded test samples were placed on the plates, heated to 160°C, allowed to rest for 15 min to relax the polymer. Each test sample was measured at maximum strains of 0.5 and frequencies from 0.01-100 s⁻¹. The storage (G') and loss (G'') moduli were then measured.

Transient shear rheology was also characterized using the ARES LS2 with the same geometry and sample heating as with SAOS tests. Five shear rates were tested: 0.01, 0.1, 0.3, 1, and 3 s⁻¹. New test samples were used for each shear rate.

Extensional rheology was characterized using a Rheometrics Melt Extensometer (RME) from Rheometrics Scientific. Test samples were heated to 160°C, allowed to rest for 15 min, and tested at one of five extension rates: 0.01, 0.03, 0.1, 0.3, 1 s⁻¹. New test samples were used for each extension rate.

Orientation evolution of CNFs in extensional flows were characterized by stretching test samples to total strains of 0, 0.1, 0.3, 1, or 3 at constant extension rates of 0.01, 0.1, or 1 s⁻¹. The samples were cut with a ultra-microtome to 200nm thickness at a 20° angle relative to the stretching direction, previously determined to be optimum [43]. A TEM microscope was used to view and characterize the average orientation of CNFs.

Chapter 3: Description of Constitutive Model

The constitutive model is composed of the following four equations:

$$\tau_{ij}^c = -p\delta_{ij} + 2\eta_s D_{ij} + \tau_{ij}^p + \tau_{ij}^{CNF} \quad (1)$$

$$\sigma \tau_{ij,m}^p + \lambda_m \frac{D\tau_{ij,m}^p}{Dt} + \frac{\alpha_m \lambda_m}{\eta_{p,m}} (\tau_{ik,m}^p \tau_{kj,m}^p) + \frac{3(1-\sigma)}{2} (a_{ik} \tau_{kj,m}^p + \tau_{ik,m}^p a_{kj}) = 2\eta_{p,m} D_{ij} \quad (2)$$

$$\tau_{ij}^{CNF} = 2[\eta_s + \eta] \varphi [AD_{kl} a_{ijkl} + B(D_{ik} a_{kl} + a_{ik} D_{kj}) + CD_{ij} + 2Fa_{ij} D_r] \quad (3)$$

$$\frac{da_{ij}}{dt} = (W_{ik} a_{kj} - a_{ik} W_{kj}) + \chi (D_{ik} a_{kj} + a_{ik} D_{kj} - 2D_{kl} a_{ijkl}) + 4C_I \Pi_D^{1/2} (\delta_{ij} - 3a_{ij}) \quad (4)$$

Equation (1) expresses the total stress in the composite. Equation (2) is the multi-mode Giesekus model which predicts the flow induced stress in the polymer, and Equation (3) expresses the flow induced stress from the CNFs. Equation (4) describes the evolution of CNF orientation during flow.

Equation (1), proposed by Azaiez [44] for fiber suspensions in viscoelastic media, expresses the total stress (τ_{ij}^c) in the composite as a sum of the pressure maintaining incompressibility (p), the stress from a Newtonian solvent ($2\eta_s D_{ij}$), from the polymer (τ_{ij}^p), and from the CNFs (τ_{ij}^{CNF}). Since no solvent was used during rheological testing in the PS/CNF composites studied, there is no solvent contribution. Transient shear viscosity is defined as

$$\eta^+ = \frac{\tau_{12}}{\dot{\gamma}}, \quad (5)$$

where $\dot{\gamma}$ is the shear rate, and extensional viscosity is

$$\eta_E^+ = \frac{\tau_{11} - \tau_{22}}{\dot{\epsilon}}, \quad (6)$$

where $\dot{\epsilon}$ is the extension rate.

Equation (2) is a multi-mode version of the model proposed by Giesekus [45] which predicts the flow induced stress in the polymer as a sum of the stresses of all the modes

$$\tau_{ij}^p = \sum_{m=1}^N \tau_{ij,m}^p, \quad (7)$$

where N is the number of modes. The upper convected derivative of τ_{ij}^p is given as

$$\frac{D\tau_{ij}^p}{Dt} = \frac{d}{dt} \tau_{ij}^p - W_{ik} \tau_{kj}^p + \tau_{ik}^p W_{kj} - D_{ik} \tau_{kj}^p - \tau_{ik}^p D_{kj}, \quad (8)$$

where W_{ij} is the skew part and D_{ij} is the symmetric part of the Eulerian velocity gradient.

For shear flow,

$$W_{ij} = \begin{bmatrix} 0 & \dot{\gamma}/2 & 0 \\ -\dot{\gamma}/2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad D_{ij} = \begin{bmatrix} 0 & \dot{\gamma}/2 & 0 \\ \dot{\gamma}/2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \tau_{ij} = \begin{bmatrix} \tau_{11} & \tau_{12} & 0 \\ \tau_{21} & \tau_{22} & 0 \\ 0 & 0 & \tau_{33} \end{bmatrix}, \quad (9)$$

and for extensional flow,

$$W_{ij} = 0, \quad D_{ij} = \begin{bmatrix} \dot{\epsilon} & 0 & 0 \\ 0 & -\dot{\epsilon}/2 & 0 \\ 0 & 0 & -\dot{\epsilon}/2 \end{bmatrix}, \quad \tau_{ij} = \begin{bmatrix} \tau_{11} & 0 & 0 \\ 0 & \tau_{22} & 0 \\ 0 & 0 & \tau_{33} \end{bmatrix}. \quad (10)$$

The remaining parameters, σ , $\eta_{p,m}$, λ_m , and α_m , are fitting parameters that describe the polymer-particle interaction and the zero strain rate viscosity of the polymer, relaxation time, and mobility factor for each mode, respectively. Of note, the appearance σ in this equation allows the coupling of τ_{ij}^p and τ_{ij}^{CNF} when $\sigma < 1$, or this coupling can be turned off by setting $\sigma = 1$. These are discussed in more detail in the following sections.

Equation (3), proposed by Tucker [46], expresses the flow induced stress contribution from the CNFs due to the traction of the polymer on the surface of the

particles, averaged over all particle orientations. In this equation, η is the viscosity of the polymer. ϕ is the volume fraction of CNFs. A, B, C, and F are shape factors which are defined below in Table 2 for various particle loading and orientation regimes where r is the aspect ratio of the CNFs given as $r = \text{length}/\text{diameter}$.

Table 2: Shape Factors for Various Concentration Regimes

Shape Factor	Dilute [47]	Semidilute Aligned [48]	Semidilute Aligned [49]	Semidilute Isotropic [49]
A	A_1	A_2	A_3	A_4
B	$\frac{6 \ln(2r) - 11}{r^2}$	0	0	0
C	2	0	0	0
F	$\frac{3r^2}{\ln(2r) - 0.5}$	0	0	0

$$A_1 = \frac{r^2}{2[\ln(2r) - 1.5]}$$

$$A_2 = \frac{r^2}{3 \ln\left(\sqrt{\left(\frac{\pi}{\phi}\right)}\right)}$$

$$A_3 = \frac{16r^2}{3 \ln\left(\frac{1}{\phi}\right)} \left[1 - \frac{\ln \ln\left(\frac{1}{\phi}\right)}{\ln\left(\frac{1}{\phi}\right)} + \frac{0.6344}{\ln\left(\frac{1}{\phi}\right)} \right]$$

$$A_4 = \frac{16r^2}{3[\ln\left(\frac{1}{\phi}\right) + \ln \ln\left(\frac{1}{\phi}\right) + 1.4389]}$$

The semi-dilute regime is defined as volume fractions within the range specified as

$$r^{-2} < \phi < r^{-1} \quad (11)$$

a_{ij} and a_{ijkl} are the second and fourth order orientation tensors that capture the average orientation of the CNFs in three-dimensions. These tensors are derived from a probability distribution of individual fiber orientations. An individual fiber can be described by a vector \mathbf{p} in three dimensional space in spherical coordinates as described by

$$\mathbf{p} = (p_1, p_2, p_3) = (\cos\phi\sin\theta, \sin\phi\sin\theta, \cos\theta) \quad (12)$$

Figure 5 demonstrates such a vector and coordinate system.

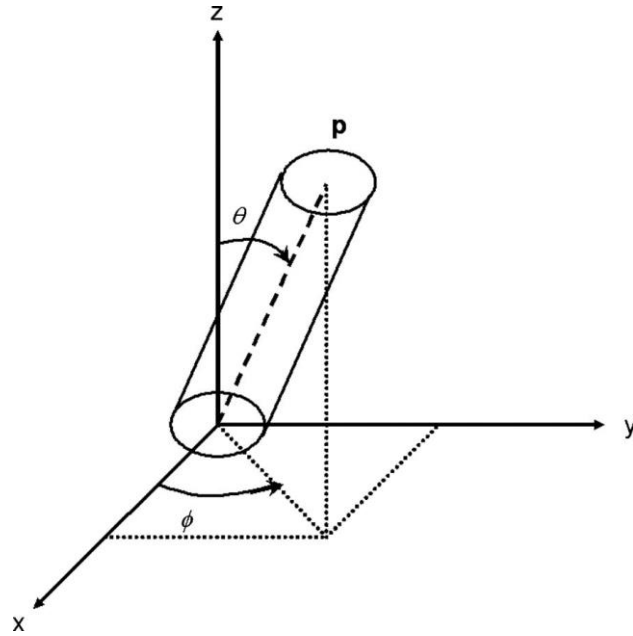


Figure 5: Coordinate System for Orientation of Single Nanofiber [10]

A probability distribution function, $\psi(\theta, \phi)$, can describe the probability, P , of finding a fiber oriented between θ_1 and $(\theta_1 + d\theta)$ and between ϕ_1 and $(\phi_1 + d\phi)$, given as [50]

$$P(\theta_i \leq \theta \leq \theta_i + d\theta, \phi_i \leq \phi \leq \phi + d\phi) = \psi(\theta, \phi) \sin \theta_i d\theta d\phi \quad (13)$$

with each fiber described by a (θ, ϕ) existing on the surface of a unit sphere and with [50]

$$\oint \psi(p) dp = \int_0^{2\pi} \int_0^{2\pi} \psi(\theta, \phi) \sin \theta d\theta d\phi = 1. \quad (14)$$

The orientation tensors can then be described over the orientation space by [50]

$$a_{ij} = \int p_i p_j \psi(p) dp \quad (15)$$

$$a_{ijkl} = \int p_i p_j p_k p_l \psi(p) dp \quad (16)$$

and by the definition of p from Equation 12,

$$a_{ij} = \begin{bmatrix} \cos^2 \theta & \sin \theta \cos \theta \sin \phi & \sin \theta \cos \theta \cos \phi \\ \sin \theta \cos \theta \sin \phi & \sin^2 \phi \sin^2 \theta & \sin^2 \theta \sin \phi \cos \phi \\ \sin \theta \cos \theta \cos \phi & \sin^2 \theta \sin \phi \cos \phi & \cos^2 \phi \sin^2 \theta \end{bmatrix}. \quad (17)$$

By averaging over a sufficiently large number of nanofibers, the second order orientation tensor can be calculated [51].

The calculation of the fourth order orientation tensor requires an approximation in terms of the second order orientation tensor. There are three common choices for this approximation: the linear closure approximation for non-aligned fibers given by [52]

$$\begin{aligned} \hat{a}_{ijkl} = & -\frac{1}{35} (\delta_{ij} \delta_{kl} + \delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk}) + \frac{1}{7} (a_{ij} \delta_{kl} + a_{ik} \delta_{jl} + a_{kl} \delta_{ij} + a_{jl} \delta_{ik} + a_{jk} \delta_{il} \\ & + a_{il} \delta_{jk}) \end{aligned} \quad (18)$$

the quadratic closure approximation for aligned fibers given by [52]

$$\tilde{a}_{ijkl} = a_{ij} a_{kl} \quad (19)$$

and the hybrid closure approximation, combining linear and quadratic closure

approximations for the entire range of non-aligned and aligned fibers given by [52]

$$a_{ijkl} = (1 - f) \hat{a}_{ijkl} + f \tilde{a}_{ijkl}, \quad f = 1 - 27 \det(a_{ij}) \quad (20)$$

The hybrid closure approximation was chosen for this research as the experiments consist of flows that orient the CNFs from a random orientation initially to an oriented state by the end of the experiment, including high degrees of orientation for extensional flows.

D_r is the rotary diffusivity due to Brownian motion defined as

$$D_r = 2C_I \Pi_D^{1/2} \quad (21)$$

where C_I is a fitting parameter describing the hydrodynamic particle-particle interaction and is discussed in more detail in the composite parameter fitting section, and Π_D is the second invariant of the symmetric part of the velocity gradient equal to $\frac{\dot{\gamma}^2}{4}$ for shear flow and $\frac{3\dot{\epsilon}^2}{4}$ for extensional flows.

The final equation in the model is Equation (4) suggested by Folgar and Tucker [53] and used by Advani and Tucker to describe the evolution of CNF orientation during flow. χ is a shape factor described by the aspect ratio as

$$\chi = \frac{r^2 - 1}{r^2 + 1}. \quad (22)$$

Chapter 4: Optimizing the Model

Parameter fitting for pure polymer

Three of the parameters discussed above, $\eta_{p,m}$, λ_m , and α_m , describe only the polymer and so could be fit to pure polymer samples. This was done by fitting the model to experimental data from pure polystyrene for transient shear and transient extensional data as well as fitting the storage and loss moduli to small-amplitude oscillatory shear (SAOS) data. The overhead program used to optimize these parameters can be found in the appendix under the “Pure Polymer: Overhead Parameter Optimization” section. From previous experimental data, it was shown that there was no difference between MB-0wt% and SC-0wt%, so the MB-0wt% was used for pure-polymer parameter fitting [14]. The storage and loss moduli were modeled with the multi-mode general linear viscoelastic model using the following equations:

$$G'(\omega) = \sum_{m=1}^N \frac{\eta_{p,m} \lambda_m \omega^2}{1 + (\lambda_m \omega)^2} \quad (23)$$

$$G''(\omega) = \sum_{m=1}^N \frac{\eta_{p,m} \omega}{1 + (\lambda_m \omega)^2} \quad (24)$$

where ω is the frequency of the oscillation. Many frequencies are tested to capture the behavior of the moduli over a range of oscillation frequencies. The model predictions were compared to SAOS experimental moduli data through the following error calculation:

$$E_{SAOS} = \sum_{m=1}^N \{ [\log_{10} G'_{exp}(\omega_i) - \log_{10} G'(\omega_i)]^2 + [\log_{10} G''_{exp}(\omega_i) - \log_{10} G''(\omega_i)]^2 \} \quad (25)$$

The program developed to optimize $\eta_{p,m}$, λ_m , and α_m for SAOS flows can be found in the appendix under “Pure Polymer: SAOS Flows”.

Model predictions for shear and extensional flows were likewise compared to experimental viscosity data through the following error calculation:

$$E = \sum_{j=1}^N [\log_{10} \eta_{exp}(t_j) - \log_{10} \eta_{model}(t_j)]^2 \quad (26)$$

where N is the number of experimental data points and t_j is the time of the j-th point.

The total error was then

$$E_{total} = E_{Extensional} + y * E_{Shear} + z * E_{SAOS} \quad (27)$$

where y and z can change the relative weighting of errors so that the error in the model's prediction of shear and SAOS are the same order of magnitude as the error in extensional predictions. Balancing the order of magnitudes of the errors yields accurate model predictions for all three flows. The programs used for transient shear and extensional flows in a pure polymer can be found in the appendix under "Pure Polymer: Transient Shear Flow" and "Pure Polymer: Transient Extensional Flow", respectively.

Additionally, the sub-programs that solved the constitutive model for these two flow fields can be found under "Constitutive Model Solver: Extensional Flow" and "Constitutive Model Solver: Shear Flow".

Since shear viscosity vs. time is typically plotted on a log-log plot, any time delay in the start-up of an experiment would be noticeable in the plot. This was accounted for by examining the experimental data on a log-log plot to see when data appeared to approach the time axis at short times. From this, it was determined that the rheometer used had a time delay of approximately 0.0095 s. By shifting all of the time data points back by this factor, a more accurate plot of experimental data at short times could be observed and compared to model predictions, both through the error minimization program and visually.

In extensional flows at low extension rates and short times, the strain is so low that large inaccuracies can occur in the experimental data due to transducer insensitivity. At these low extension rates and strains, it is known that extensional viscosity follows the linear viscoelastic plateau so these inaccuracies can be correctly labeled as transducer error. However, when fitting a model to this data, these large deviations make minimizing the error between model and experimental data more difficult, and can even lead away from optimal parameter values. Thus, the experimental data for the two lowest strain rates were truncated at short times, leaving the experimental data at higher times which was free of these large inaccuracies. This gave a smoother curve fit the model to and resulted in a far more accurate optimization of model parameters.

A program was written in MATLAB to solve the model, and through the use of a constrained minimization function, “fmincon”, minimize the error between experiment and model for all three flow fields by changing $\eta_{p,m}$, λ_m , and α_m . Fmincon is a MATLAB function that finds the minimum of a constrained nonlinear multivariable function, in this case, the error between nonlinear functions and experimental data. The parameter design space was examined by the minimization function through the use of the “Interior-point” algorithm. This algorithm is the standard one used by MATLAB and is useful for large, sparse problems and small, dense problems. Additionally, this algorithm is capable of recovering from divergent results, and all constraints are satisfied with each iteration [54]. The number of modes was incrementally increased from one until the model could accurately capture the behavior of the polymer and additional modes provided no significant improvements.

Parameter fitting for composite

The remaining two parameters, σ and C_I , describe effects due to nanofibers in a composite and thus were fit to experimental composite data. A similar approach to fitting these parameters using error analysis as described for the pure polymer was initially employed but did not yield ideal results due to a gross over-prediction of viscosity. This had been observed in this model before by Kagarise [10], and the only solution to bring the model prediction within the range of experimental data was to scale the experimentally measured aspect ratio, thus making an “effective” aspect ratio. An additional motivation behind this scaling factor was that an agglomerate of fibers would have a reduced aspect ratio below that of a single fiber. Therefore, this scaling factor might be able to capture the amount of agglomeration in the composites.

In this research, effective aspect ratio was made into another fitting parameter for composite data defined as

$$r_{effective} = r_{exp}/h \quad (28)$$

where h is the scaling factor. However, introducing this parameter now caused the optimization of aspect ratio to dominate the optimization process and mask the effect σ and C_I . Therefore, a new method for optimizing these three parameters was created.

Since C_I is the only parameter that appears in the Equation 4, the orientation evolution of nanofibers, it can be fit to the steady-state orientation of nanofibers, as suggested by Kagarise [10]. Steady-state orientation was measured only for extensional flows at four strains since a high degree of orientation is achieved in this type of flow. Equation 4 was set to zero, and the error between experimental data at three strain rates

(0.01, 0.1, and 1 s⁻¹) and model predictions for the component of the orientation tensor in the direction of flow, a_{11} , given by

$$E_{a_{11}} = [a_{11,exp} - a_{11,model}]^2 \quad (29)$$

was minimized to yield the optimal value for C_I . The same function, `fmincon`, and algorithm, Interior-point, that were used in the pure polymer optimization were also used in this error minimization. This program can be found in the appendix under “Composite Parameter Optimization: C_I ”.

To fit σ and $r_{effective}$ the major effects these parameters have on model predictions needed to be examined. This was done by changing one parameter while fixing the other and minimizing the error between the resulting viscosity predictions of the model and experiment using the same algorithm as before. These results can be seen below in Figure 6 and Figure 7.

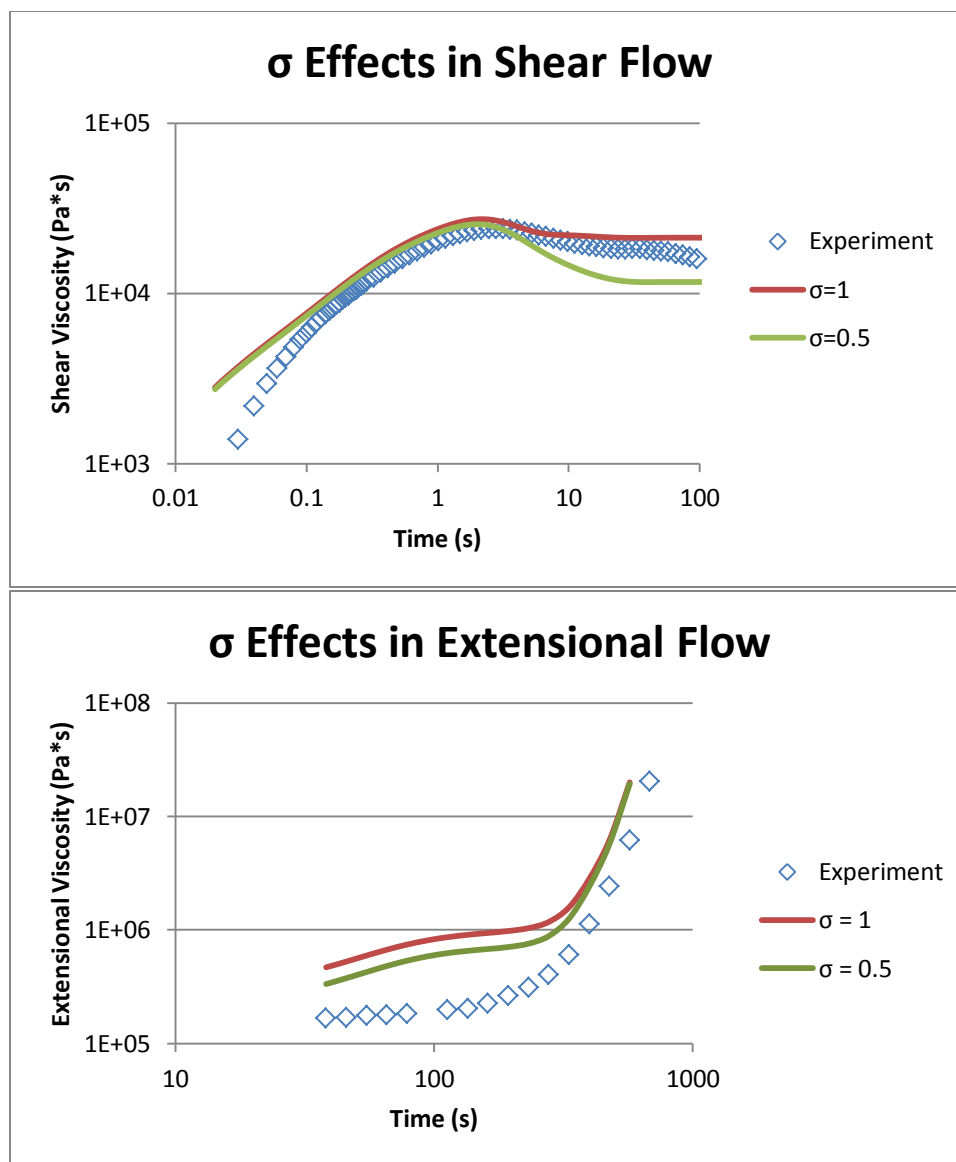


Figure 6: Effect of σ in (a) shear and (b) extensional flow; other parameters fixed:

$$C_l = 0.005, r = 40, 2wt\%$$

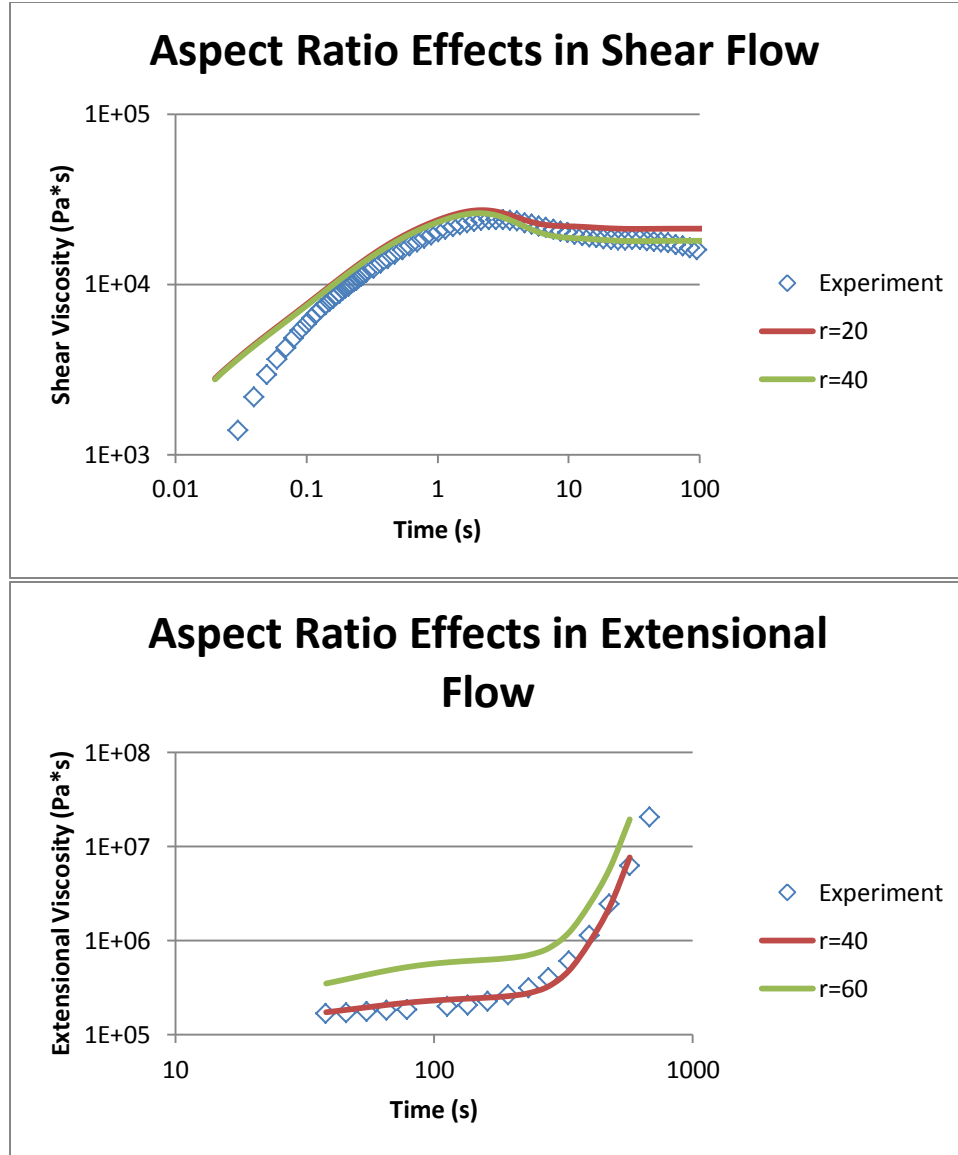


Figure 7: Effect of $r_{effective}$ in (a) shear and (b) extensional flow; other parameters

fixed: $C_I = 0.005, \sigma = 0.75, 2wt\%$

From this analysis, the following optimization scheme was created. Since aspect ratio has a large effect on the order of magnitude of the viscosity prediction, the peak viscosity overshoot in experimental shear data and the linear viscoelastic plateau (LVP) in experimental extensional data from 1 to 100 seconds were defined as characteristic

points to fit the model to. The error of the shear characteristic point between experiment and model as well as the error of the extensional characteristic point between experiment and model was minimized by optimizing $r_{effective}$ with the following equation

$$E = [\log_{10} \eta_{max,exp} - \log_{10} \eta_{max,model}]_{shear}^2 + z * [\log_{10} \eta_{LVP,exp} - \log_{10} \eta_{LVP,model}]_{extension}^2 \quad (30)$$

where z can change the relative weighting of shear to extensional error so that both are the same order of magnitude, thus equally weighting the contribution to error from both flows. This result was then used to optimize the value of σ . Since σ mainly affects the amount of shear thinning, the error between the experimental amount of shear thinning the model's prediction was minimized by optimizing σ with the following equation

$$E = [\eta_{SS,exp} - \eta_{SS,model}]^2 \quad (31)$$

This process was then used iteratively to converge to optimal values for $r_{effective}$ and σ . These programs can be found in the appendix under “Composite Parameter Optimization: Aspect Ratio Scaling Factor” and “Composite Parameter Optimization: σ ”. Once C_I , σ , and effective aspect ratio were fit, the model predictions could be made and compared against experimental data. Programs for generating the model predictions can be found in the appendix for the three composites, all under sections starting with “Composite Model Predictions”.

SAOS experimental data could also be fit by this model. Because of the addition of CNFs, the multi-mode Maxwell model for the storage and loss moduli no longer applies. However, an alternative method can be applied as suggested by Kremer [40].

The strain wave applied in SAOS experiments can be described by

$$\gamma(t) = \gamma^0 \sin \omega t \quad (32)$$

The shear rate is the derivative of the strain wave given by

$$\frac{d\gamma}{dt} = \dot{\gamma}(t) = \gamma^0 \cos \omega t \quad (33)$$

The stress wave can then be written as contributions from the in-phase and out-of-phase stresses which are functions of the storage and loss moduli, respectively. The stress wave is given by

$$\tau_{12}^{Gs} = G'(\omega)\gamma^0 \sin \omega t + G''(\omega)\gamma^0 \cos \omega t \quad (34)$$

The moduli are fit in two steps. First, the model is solved using Equation 33 in place of a constant strain rate to predict the stress wave for a given frequency. This method can be seen in the program under the section “Composite Model Predictions: Modeling the Stress Wave in SAOS Flow” in the appendix. Then using an initial guess for G' and G'' , Equation 34 is solved, and the error between these two stress waves is minimized where error is defined as

$$E = \sum_{j=1}^N [\tau_{12}^c(t_j) - \tau_{12}^{Gs}(t_j)]^2 \quad (35)$$

where N is the number of experimentally tested frequencies. This process is repeated for all experimentally tested frequencies giving $G'_{model}(\omega)$ and $G''_{model}(\omega)$. This can be seen in the appendix under “Composite Model Predictions: Solving the Constitutive Model for SAOS Flow” for the model solving and under “Composite Model Predictions: Fitting G' and G'' ” for the fitting of moduli to the oscillatory stress wave.

Divergence

Previous work by Kremer [40] found that when σ is allowed to be less than one, at certain values of λ and α the model prediction for shear viscosity would diverge to

negative infinity. In this work, the same problem was also seen in extensional viscosity, but with the model prediction diverging to positive infinity. The way he dealt with this problem was to map out the design space of σ , λ , and α values that lay on the boundary of the model converging/diverging by testing values of these three parameters within the following range: $0.1 \leq \alpha \leq 1$, $0.1 \leq \lambda \leq 10$, and $0.6 \leq \sigma \leq 1$. His plot can be seen below in Figure 8.

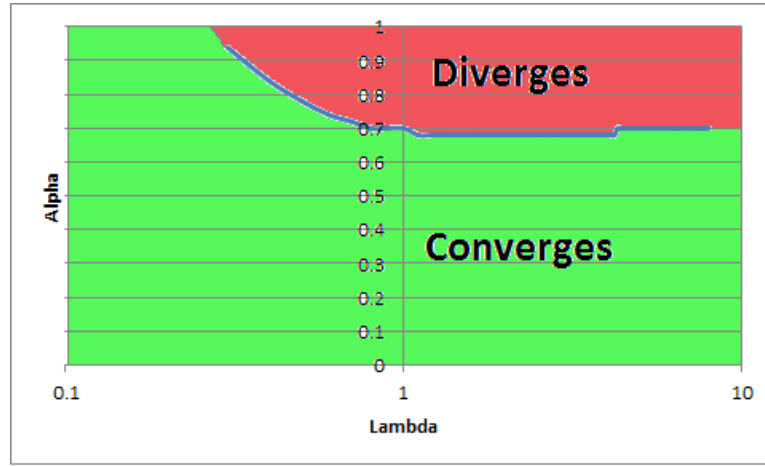


Figure 8: Convergence area for λ and α values at $\sigma = 0.6$, 5wt% CNFs, and $\dot{\gamma} = 1s^{-1}$ [40]

As σ decreases, the diverging boundary shifts vertically downward in α . Additionally, larger CNF concentrations, aspect ratios, and strain rates increase the area of divergence. Therefore, by specifying the smallest value of σ , largest concentration of CNFs, largest aspect ratio, and largest strain rate of interest, the extreme points of the diverging boundary can be defined. Once these are defined, by limiting the error optimization program to only look in the converging area of α and λ , pure polymer parameters can be fit that will not cause the model to diverge when modeling composites in the future. A

program for defining the convergence boundary can be found in the appendix under “Determining Boundary of Convergence for σ , λ , and α ”.

Since this study was examining larger aspect ratios, lower CNF loadings, and extensional flows, this process of mapping out the area of divergence was repeated. However, the range on σ was increased 0.5 to allow for any increased polymer-particle interaction that may occur with the new preparation methods. The new convergence area used in this research can be seen in Figure 9.

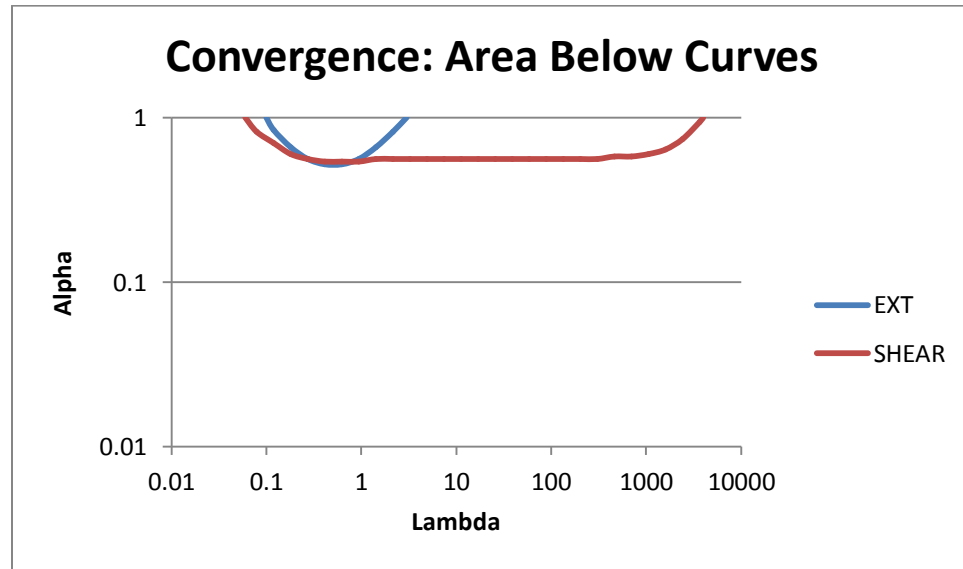


Figure 9: New convergence area for λ and α values at $\sigma = 0.5$, 2wt% CNFs, and

$$\dot{\epsilon} = 1s^{-1}, \dot{\gamma} = 3s^{-1}$$

This diverging can also be displayed as a surface over the range of σ values, where the set of parameter values inside the surface diverge. Such a surface for extensional and

shear flows can be seen in

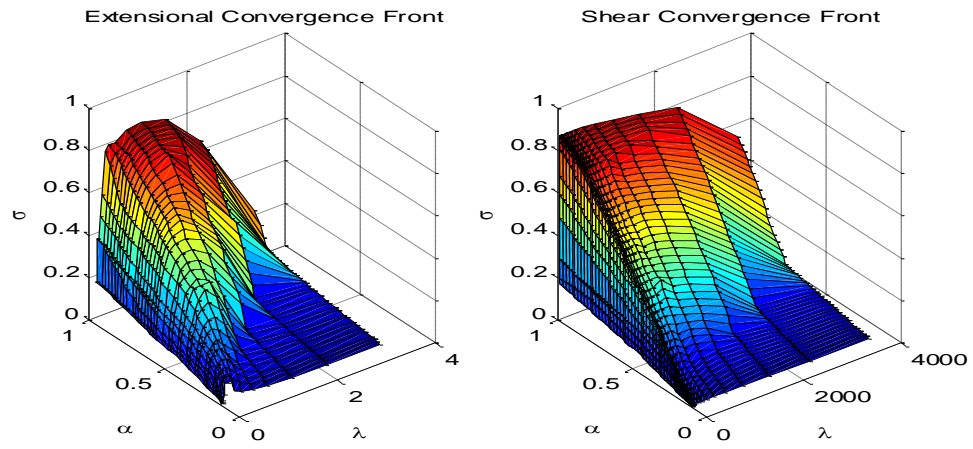


Figure 10.

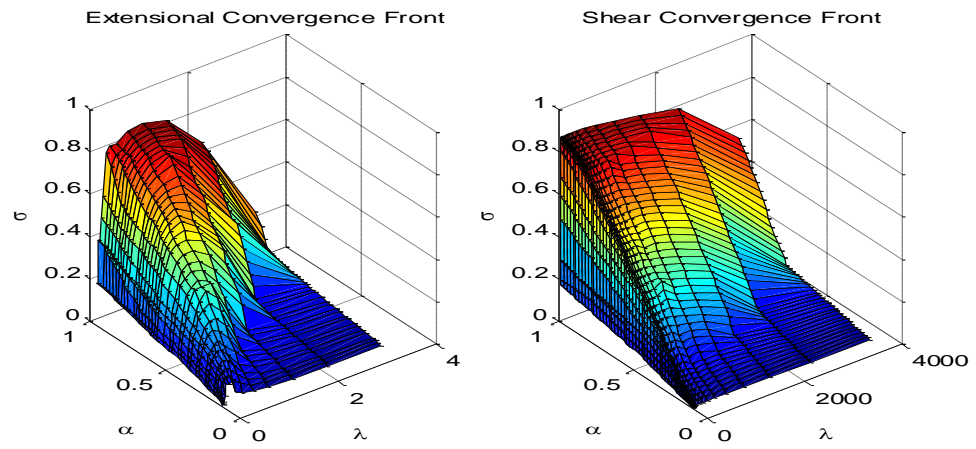


Figure 10: Divergence Surface for (a) extensional flow (b) shear flow

Chapter 5: Results and Discussion

Pure polymer

Modes and Optimized Parameters

Previous research had used 5 modes, but it was found that using 6 modes allowed the strain-hardening behavior in extensional flow, especially at low extension rates, to be captured far more accurately than with only 5 modes. Adding a seventh mode hardly increased the accuracy of the model and led to two modes having very similar relaxation times, suggesting the additional mode was not needed. Because using more modes also increases the complexity of solving the equations and optimizing the parameters, 6 modes were determined to be optimal.

The error minimization technique and program described above for pure polymers were used to optimize the values for $\eta_{p,m}$, λ_m , and α_m . These values are shown in Table 3

Table 3: Optimized Parameter Values for Pure Polymer

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
λ_m (sec)	0.0146	0.236	3.21	34.4	6390	116000
$\eta_{p,m}$ (Pa*s)	1560	10500	30800	9860	9310	40700
α_m	0.819	0.560	0.560	0.0786	0.0252	0.00184

The error weighting parameters were found to be $y = 0.1$ for shear and $z = 100$ for SAOS so that all error contributions were of the same order of magnitude.

The values of λ_m , the relaxation time for mode m , increase by an order of magnitude from mode to mode as is common for polymers, except for the last two modes. Constraining these modes to follow the order of magnitude pattern decreased the accuracy of results, and since a seventh mode had already been shown to not improve accuracy, it was concluded that a relaxation time of order 10^2 was unnecessary. The final relaxation time is of great interest since it is such a long relaxation time. When this value was constrained to a range with lower values, the optimization kept running up against the constrained boundary, suggesting a larger relaxation time would be optimal. These constrained optimization additionally did not provide as accurate results as the unconstrained optimal, particularly in extensional viscosity predictions at large times. Therefore, despite the abnormally large value, it was determined that this relaxation time was important to the accuracy of the model predictions.

The values of $\eta_{p,m}$, the zero strain viscosity of the polymer, are generally of order 10^4 magnitude. The first mode shows a small η_p because it is the dominant mode at short times when the viscosity of the polymer is small. The last mode is also of note, displaying the largest zero strain viscosity, and due to the large relaxation time of the mode, this zero strain viscosity is important in capturing the strain hardening behavior at long times.

The values of α_m , the mobility factor, also display a range of magnitudes. It was observed that smaller values of α lead to more abrupt changes in the viscosity during strain hardening in extensional flows. It therefore makes sense that the last mode with

the longest relaxation time has a very small value of α , allowing the polymer to strain harden quickly as seen in the experimental extensional viscosity at long times.

Additionally, modes 4 and 5 also have relatively smaller values of α than the first two, and these relaxation times occur near the experimental times when strain hardening occurs in the smaller extension rates ($0.1, 0.03, 0.01 \text{ s}^{-1}$) and therefore influence the prediction of the model.

Small-Amplitude Oscillatory Flow Predictions

The model prediction for storage and loss moduli can be seen in Figure 11.

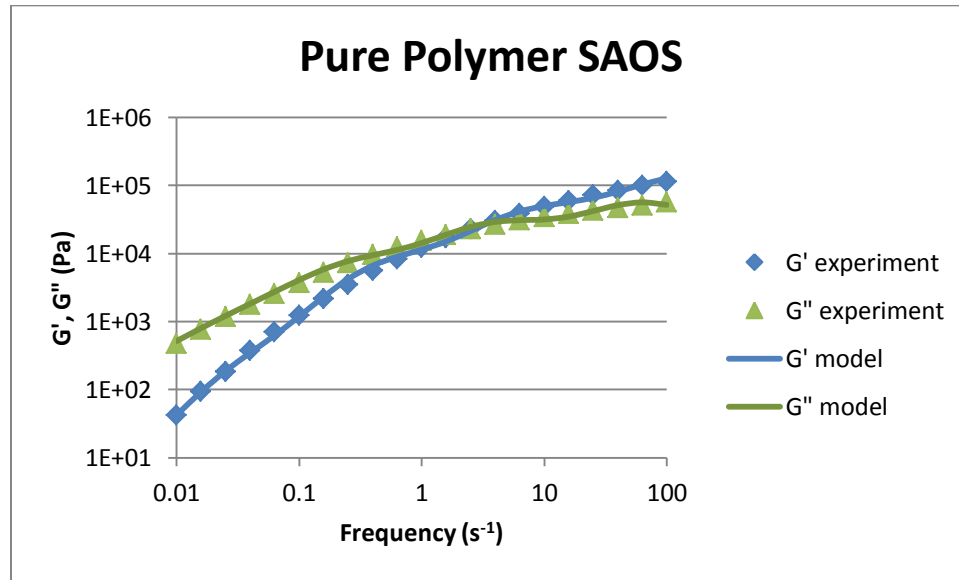


Figure 11: Pure polymer predictions for storage and loss moduli

The experimental data is accurately captured by the model. G' and G'' fitting did not have any notable difficulties. What was important in G' and G'' fitting was an appropriate value of the relative error weighting parameters, y and z , so that the error due to SAOS predictions and due to extensional and shear predictions were of the same order

of magnitude. By achieving this, the optimization was equally weighted amongst all three flow fields and gave accurate predictions for all three.

Shear Flow Predictions

The model prediction for shear viscosity can be seen in Figure 12.

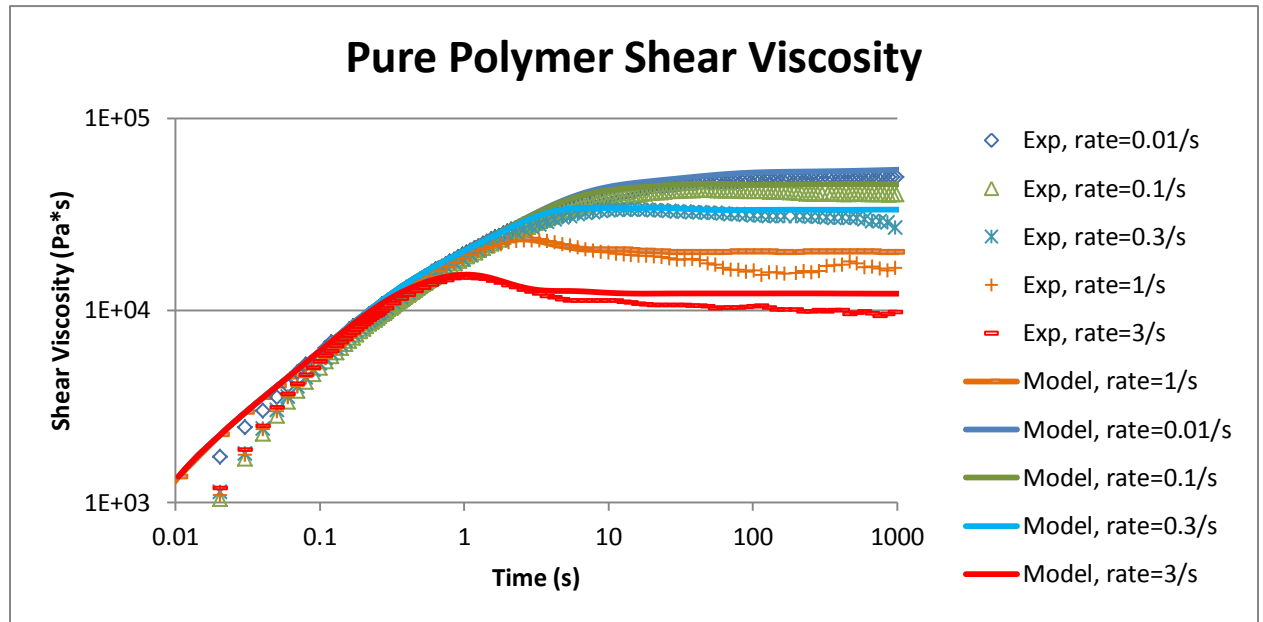


Figure 12: Pure polymer prediction for shear viscosity

Shear viscosity data was accurately fit by the model over five shear rates. A particular problem with fitting the shear data was the delay time of the rheometer. On a log-log plot, this delay time caused the experimental data to look like it was dropping to zero viscosity at a non-zero time; however, the starting point for the model was at 0 seconds and 0 Pa*s, and thus showed an over-prediction of viscosity at very short times. By shifting the experimental data by the delay time of the rheometer, 0.0095 s, this over-prediction disappeared, showing the model to be accurate even at short times.

Extensional Flow Predictions

The model prediction for extensional viscosity can be seen in Figure 13.

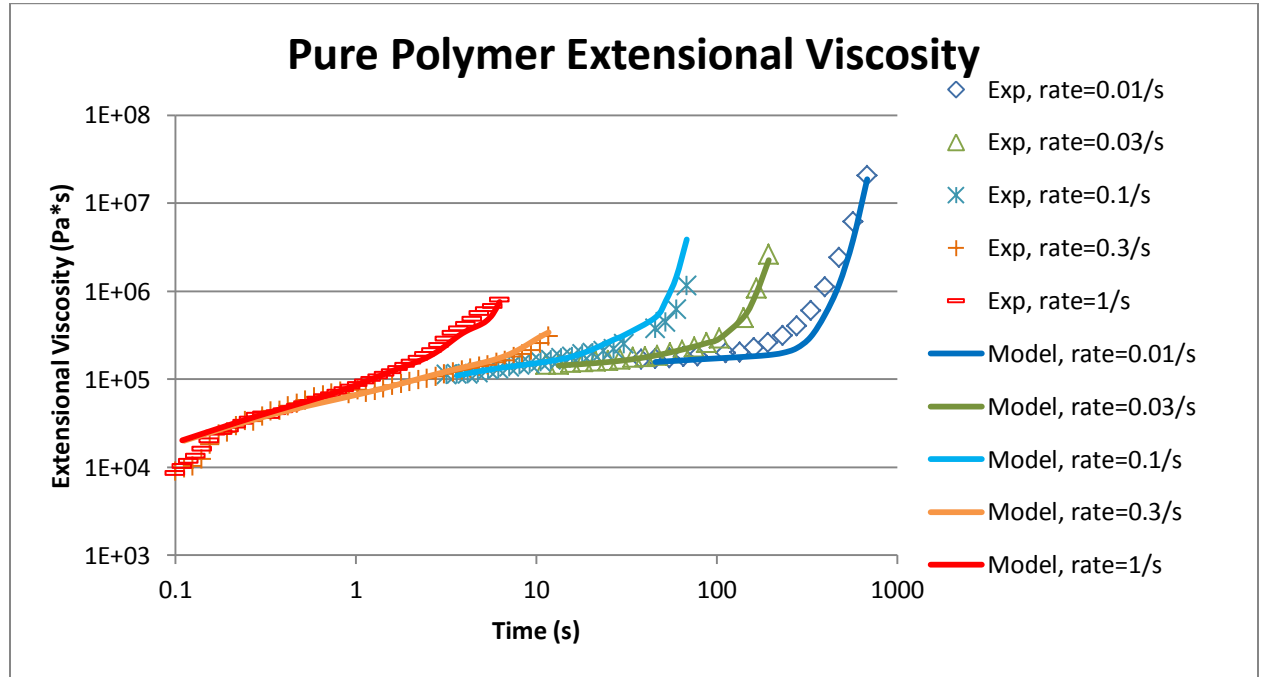


Figure 13: Pure polymer prediction for extensional viscosity

As discussed above, the range of parameter values amongst the six modes allow an accurate model prediction for extensional viscosity over all five extension rates.

Especially difficult to capture was the experimental data at the smallest extension rate, 0.01 s^{-1} . This was improved over past research by allowing an additional mode with a very long relaxation time, mode 6.

Composite

Shape factor

Previous work by Kagarise found 2wt% composites to be in the semi-dilute regime and suggested the use of A_2 in the semidilute-aligned regime [10]. The three different shape factors for semidilute regimes, A_2 , A_3 , and A_4 , were re-examined in the context of this research since an additional preparation technique and fiber type were being examined and as an attempt to improve the accuracy of the model. The predictions of both shear and extensional rheology using a range of σ and C_I values for each shape factor were compared. This comparison can be seen in Figure 14.

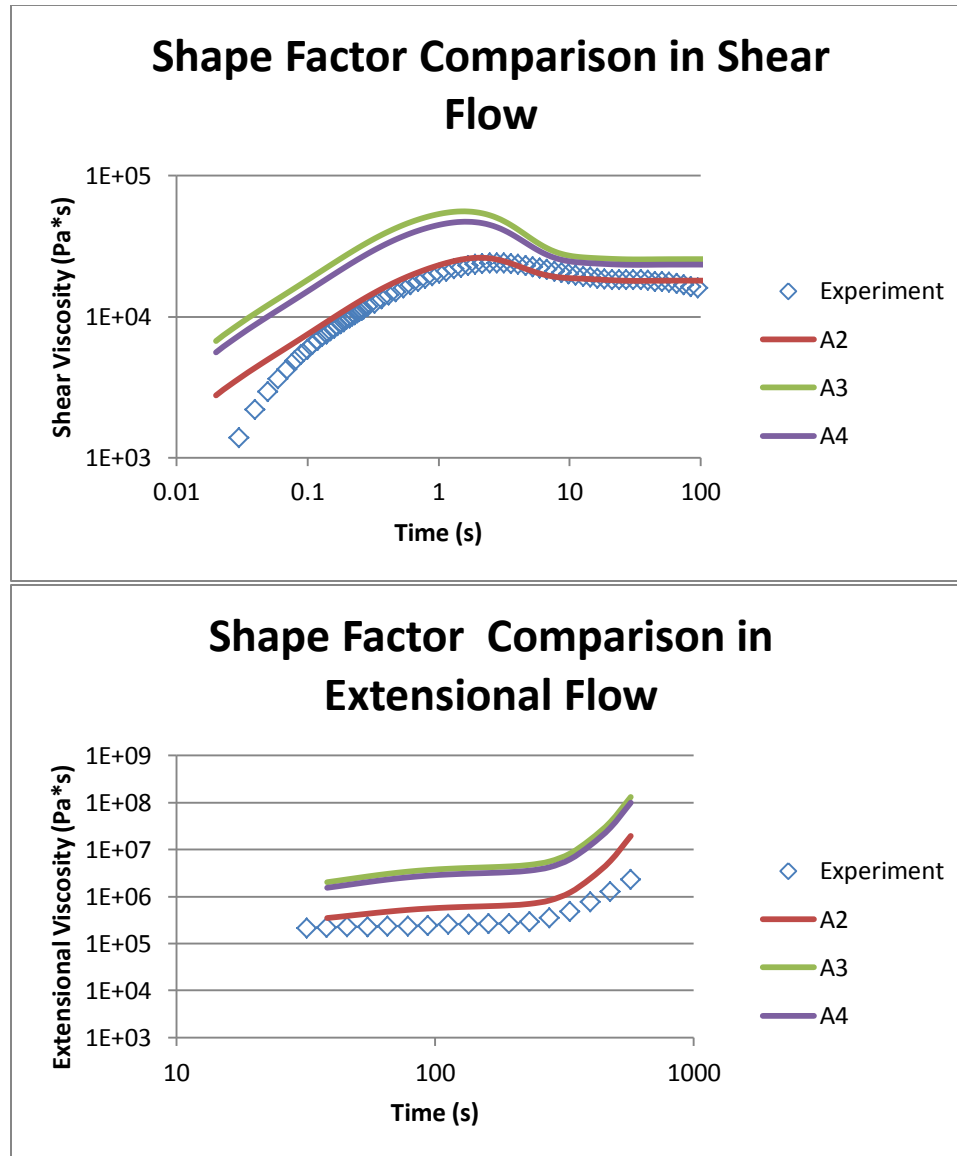


Figure 14: Shape Factor Comparison (a) shear flow (b) extensional flow

As can be seen from Figure 14, A_2 was found to be the best choice in agreement with previous studies by Kagarise of MB composites alone [10].

Optimized Parameters

The methods and programs described above for composites were used to optimize the values for σ , C_I and $r_{effective}$ for MB 2wt%, MB-HHT 2wt%, and SC 2wt% composites. The relative error weighting factor was chosen to be $z = 0.01$ so that the error contributions from shear and extensional model predictions were of the same order of magnitude. The values for the optimized parameters are shown in Table 4.

Table 4: Optimized Parameter Values for Composites

	MB 2wt%	MB-HHT 2wt%	SC 2wt%
C_I	0.0306	0.0301	0.0499
σ	0.810	0.544	0.500
$r_{effective} = r_{exp}/h$	53 / 1.84	44 / 1.45	69 / 1.76

The effects of these parameters are discussed further in the discussion of the model predictions, but a brief overview is discussed here.

C_I , the particle-particle interaction parameter, is very similar between MB and MB-HHT composites but differs more in SC composites. A larger value of C_I suggests more interaction between the particles which makes sense since SC composites have the longest fibers. Additionally, a larger value of C_I results in a lower steady-state orientation of CNFs in extensional flows.

σ , the polymer-particle interaction parameter, ranges more broadly over the three different composites. Since $\sigma = 1$ implies no interaction, smaller values of σ signify larger interactions. These interactions likely are due to two effects: CNF surface

chemistry differences and physical interaction between fibers and polymer molecules. SC composites which had the longest CNFs displays the smallest value of σ , suggesting that longer fibers interact more with polymer molecules. Such interactions could be entanglement of the fibers with polymer molecules or that longer fibers “see” more polymer molecules and can thus interact with more polymer molecules than shorter fibers. However, this trend of longer fibers having a lower value of σ is reversed when comparing MB to MB-HHT composites. This could suggest a surface chemistry difference between HHT and O CNFS. An additional study of SC-HHT composites may be able to define how values of σ capture both types of polymer-particle interactions. This research also did not characterize surface chemistry so the effect of surface chemistry differences between fibers cannot be quantitatively described with σ as found in this research. Additionally, the lower bound of tested σ values was 0.5, so further investigation into lower σ values is necessary to better define the difference between SC and MB composites. Because of the diverging issue, this would involve restricting the design space on α and λ to avoid diverging at lower σ values.

h , the effective aspect ratio scaling factor, is similar amongst all three composites. MB-HHT had the lowest value of h suggesting the least agglomeration of fibers. Since HHT fibers likely have a different surface chemistry as mentioned above, this may lead to less agglomeration of fibers than O-CNFs. Since σ suggests more polymer-particle interactions for the HHT-CNFs, there is likely a stronger affinity between the polymer and HHT-CNFs than with O-CNFs which would lead to a better dispersion of the HHT-CNFs than O-CNFs. Additionally, SC and MB composites displayed only a small difference in h , suggesting similar amounts of agglomeration with SC having slightly

less. However, this result is unexpected as the solvent-casting preparation method is known to not distribute fibers as well as melt-blending; thus, in SC composites, a larger scaling factor should be expected than in MB. These results would suggest the affinity between fibers and the polymer has more of an impact on the amount of agglomeration than the preparation method; however, the amount of agglomeration in all composites should be determined experimentally to justify the use of an effective aspect ratio, to determine relative amounts of agglomeration between composites, and even to quantitatively determine what an effective aspect ratio could be by the average amount of fibers in an agglomerate.

Small-Amplitude Oscillatory Shear Flow Predictions

Model predictions for SAOS flow can be seen in Figure 15.

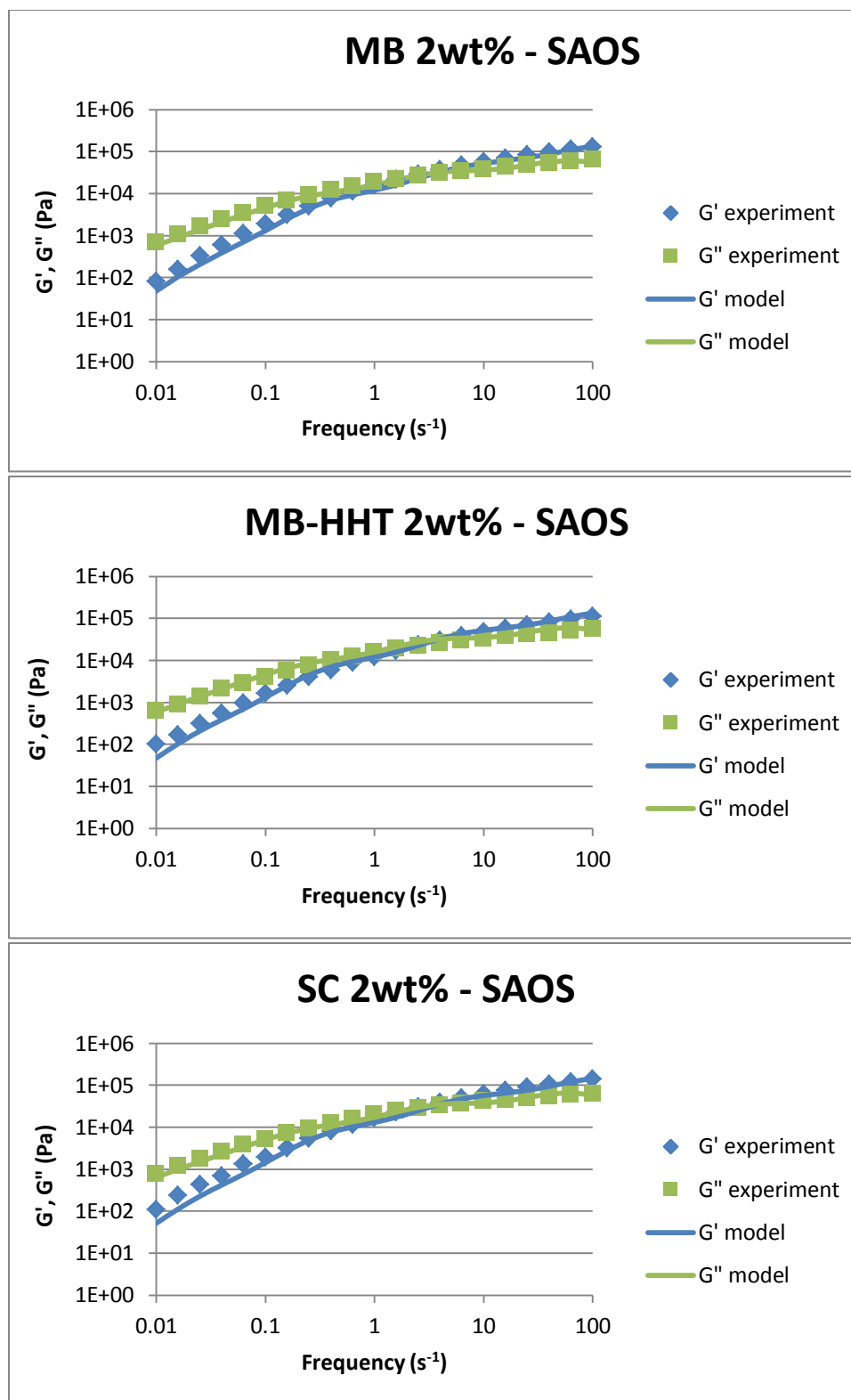


Figure 15: Moduli Predictions for: (a) MB 2wt%, (b) MB-HHT 2wt%, (c) SC 2wt%

These plots show that the model can accurately predict the moduli for all three composites. This also further verifies Kremer's method of fitting to moduli data [40].

Shear Flow Predictions

Model predictions for shear flow can be seen in Figure 16.

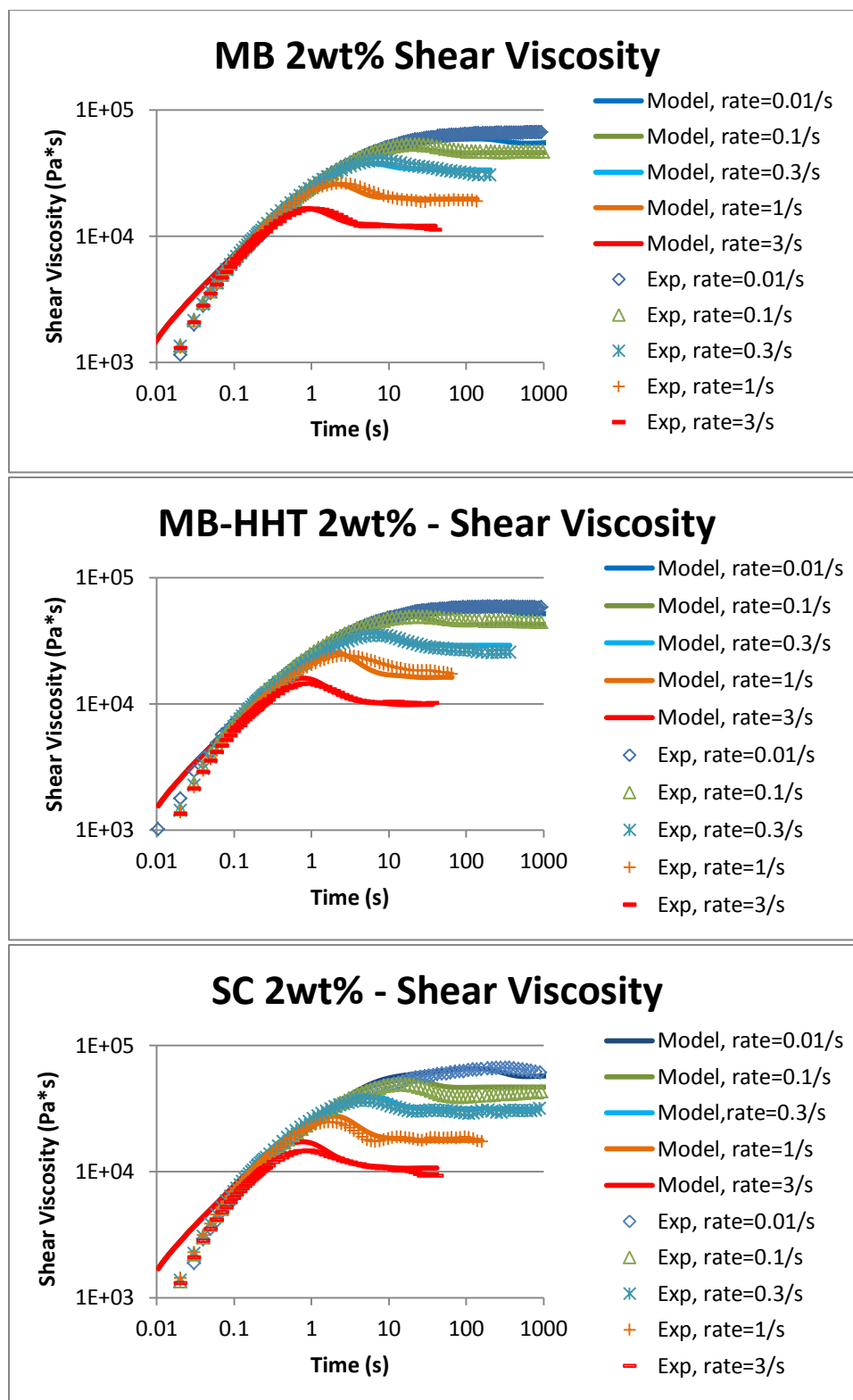


Figure 16: Shear Viscosity Predictions for: (a) MB 2wt%, (b) MB-HHT 2wt%, (c) SC

2wt%

The model accurately captures the shear viscosity behavior for all three composites. In particular, by using the parameter optimization method discussed above for h and σ , the model predictions are all of the correct order of magnitude, heavily influenced by h , and possess the proper amount of shear thinning, influenced by σ . The largest inaccuracy in the predictions is at short times, when the model slightly over-predicts the shear viscosity. Accounting for rheometer start-up delay eliminated this problem in pure polymer predictions and improved composite predictions, but a small inaccuracy is still visible particularly in MB and SC composites. The start-up time could be re-examined in the context of the composites alone to verify the delay is not larger, but since this is such a minor issue and since the model is accurate everywhere else, this is likely not necessary to improve.

Extensional Flow Predictions

Model predictions for extensional flow can be seen in Figure 17.

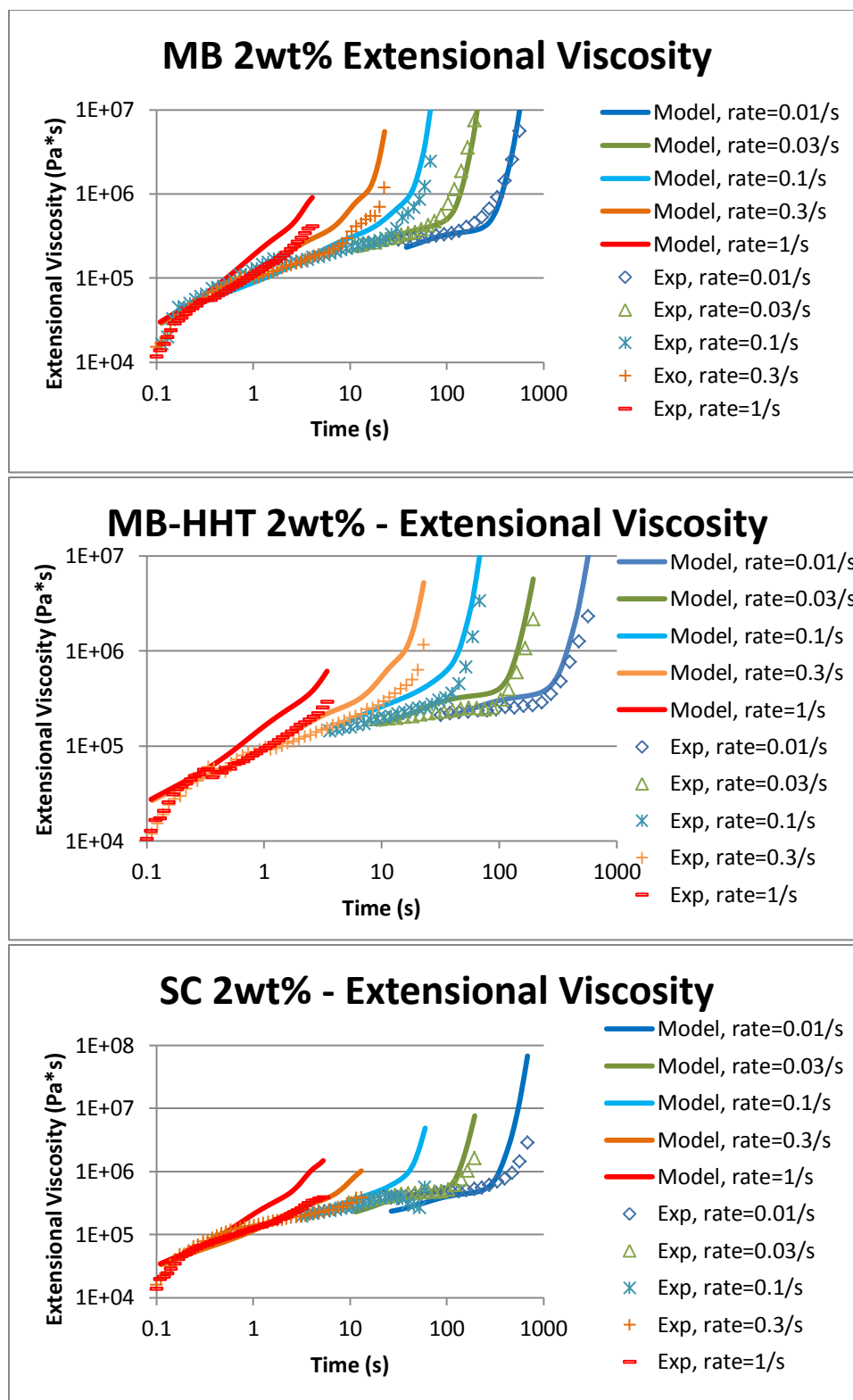


Figure 17: Extensional Viscosity Predictions for: (a) MB 2wt%, (b) MB-HHT 2wt%, (c)

SC 2wt%

The model predictions are fairly accurate for extensional flows as well, especially for MB and MB-HHT composites. All composites experience a slight over-prediction in extensional viscosity at larger extension rates, but SC composites also experience an early rise in strain hardening at the lowest two extension rates. Since α is responsible for this behavior, this could suggest that the addition of CNFs somehow affects the mobility factor of the polymer molecules. Future work could examine if fiber interactions with polymer molecules could influence values of α and provide another source of coupling between polymer and CNF behavior.

Orientation Tensor Predictions

Model predictions for orientation evolution for a_{11} , the component of the orientation tensor in the direction of flow, can be seen in Figure 18.

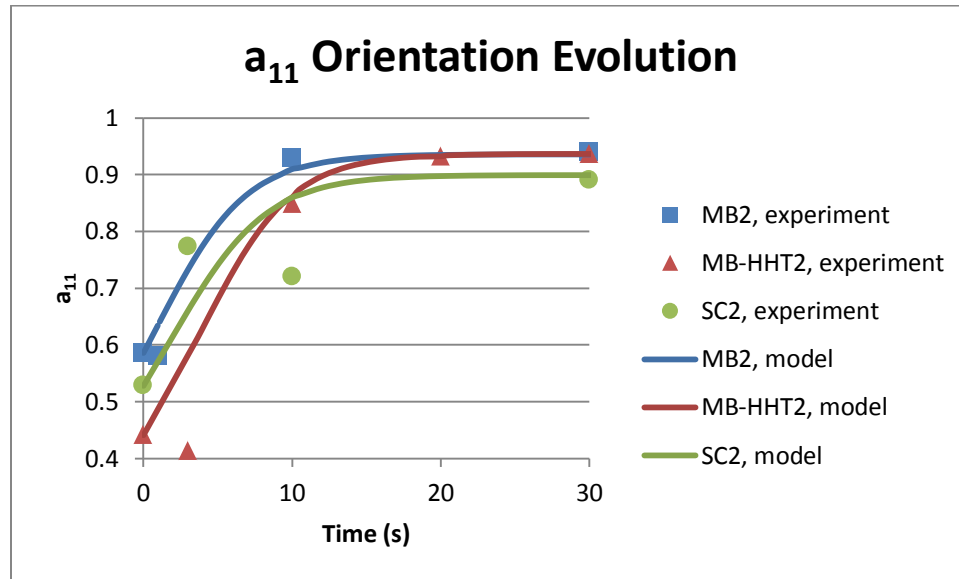


Figure 18: Orientation Evolution of a_{11} , $\dot{\epsilon} = 0.1s^{-1}$

The model predicts the steady-state orientation of CNFs and the general trend of orientation evolution well. However, the experimental data shows significant error such as the second data point for MB-HHT2 decreasing from the first and the third data point for SC2 decreasing from the second, not to be expected experimentally as a_{11} should continuously increase to the steady-state value. Repeated experiments examining the orientation evolution more closely would allow for better comparison to the model prediction. For the purpose of steady-state orientation, though, the model is accurate, and the evolution of orientation in the model seems to be reasonable.

Chapter 6: Conclusions and Future Work

Conclusions

This research was focused on modeling the rheological behavior of polystyrene-carbon nanofiber (PS-CNF) composites. Three different flow fields, extensional, shear, and small-amplitude oscillatory shear (SAOS), were examined. The model used 6 six modes and contains 3 parameters in each mode that only describe pure polymer behavior as well as three additional parameters that are used in composite modeling.

Experiments in the three flow fields for a range of oscillatory frequencies from 0.01 to 100 s⁻¹, 5 shear rates, and 5 extension rates were examined for a pure polymer, and data from these experiments were used to fit the parameters pertaining to the behavior of the polymer: zero shear viscosities, relaxation times, and mobility factors of the polymer.

PS-CNF composites were then prepared using two different preparation methods, melt-blending and solvent-casting, as well as two different types of CNFs, regular CNFs with no treatment (O-CNFs) and high-heat treated CNFS (HHT-CNFs). Three types of composites were examined all at 2 wt% loading of CNFs: melt-blended with O-CNFs (MB), melt-blended with HHT-CNFs (MB-HHT), and solvent-casted with O-CNFs (SC). The same three flow fields as the pure polymer were examined for these composites. The remaining three parameters were fit for all three composites: particle-particle interaction parameter, polymer-particle interaction parameter, and scaling factor for effective aspect ratio. Orientation evolution of CNFs were also examined and modeled.

From these results, an accurate model prediction was obtained for PS-CNF composites at 0 and 2 wt%, prepared different ways and with different CNFs, and for

three different flow fields at different strain rates. The relative values of parameters between composites can additionally give insight into the different behavior observed experimentally. Finally, a method for fitting polymer-nanoparticle composites has been further developed so that accurate models can be created.

Future Work

There is certainly room for future work. Higher loadings of CNFs have been examined by others in this research group but have not been studied as the composites in this research have. With the developments of this research and study of composites under extensional, shear, and SAOS, composites with higher CNF loadings could be studied in more detail and accuracy. In this research, an effective aspect ratio was used with the motivation that agglomerations were present in the composites. An investigation into this assumption is certainly a next step and would either verify this assumption or raise new questions into the reason for viscosity over-predictions in the model. A more detailed analysis into the evolution of the orientation tensor experimentally would also aid in verifying the accuracy of the model predictions of orientation evolution. Finally, expanding the parameter space of λ , α , and σ , to allow for model convergence at values of $\sigma < 0.5$ should be investigated. Optimization of σ for SC composites led to the boundary value of $\sigma = 0.5$ suggesting lower σ values to provide more accurate model predictions.

References

- [1] Xu, Jianhua; Chatterjee, Swaroop; Koelling, Kurt W.; Wang, Yingru; Bechtel, Stephen E. Shear and extensional rheology of carbon nanofiber suspensions. *Rheol Acta* **2005**, 44: 537–562
- [2] Shaffer, Milo S.P.; Sandler, Jan K.W. Carbon Nanotube/Nanofibre Polymer Composites.
- [3] Thostenson, Erik T.; Ren, Zhifeng; Chou, Twu-Wei. Advances in the science and technology of carbon nanotubes and their composites: a review. *Composites Science and Technology* **2001**, 61(13): 1899-1912.
- [4] Thostenson, Erik T.; Li, Chunyu; Chou, Twu-Wei. Nanocomposites in context. *Composites Science and Technology* **2005**, 65 (3-4): 491-516.
- [5] MacDonald, Rebecca A.; Laurenzi, Brendan F.; Viswanathan, Gunaranjan; Ajayan, Pulickel M.; Stegemann, Jan P. Collagen-carbon nanotube composite materials as scaffolds in tissue engineering. *Journal of Biomedical Materials Research* **2005**, 74A (3): 489-496.
- [6] Endo, Morinobu; Kim, Yoong A.; Ezaka, Masay; Osada, Koji; Yanagisawa, Takashi; Hayashi, Takuya; Terrones, Marucio; Dresselhaus, Mildred S. Selective and efficient impregnation of metal nanoparticles on cup-stacked-type carbon nanofibers. *Nano Letters* **2003**, 3(6): 723-726.
- [7] Xu, Jianhua; Wang, Yingru; Kagarise, Christopher; Koelling, Kurt W.; Bechtel, Stephen E. Shear Rheology of Nanofiber/Polymer Melt Composites.

- [8] Wang, Yingru; Xu, Jianhua; Bechtel, Stephen E.; Koelling, Kurt W. Melt shear rheology of carbon nanofiber/polystyrene composites. *Rheol Acta* **2006**, 45: 919–941.
- [9] Kagarise, Christopher; Miyazono, Koki; Mahboob, Monon; Koelling, Kurt W.; Bechtel, Stephen E. A constitutive model for characterization of shear and Extensional Rheology and flow induced orientation of carbon nanofiber/polystyrene melt composites. *The Society of Rheology, Inc. J. Rheol.* **2011**, 55(4), 781-807 July/August.
- [10] Kagarise, Christopher D. Rheological Characterization and Modeling of Micro- and Nano-Scale Particle Suspensions. Diss. Ohio State University, **2009**. OhioLINK. Web. Apr. 2013.
- [11] Giannelis, E.P. Polymer layered silicate nanocomposites. *Adv. Mater.* **1996**, 8 (1), 29-35.
- [12] Theng, B.K.G. Formation and properties of clay-polymer complexes. Elsevier **1979** (Amsterdam).
- [13] Bhattacharya, Sati N.; Gupta, Rahul K.; Kamal, Musa R. *Polymeric Nanocomposites – Theory and Practice*. Hanser Gardner Publications, Cincinnati. 15.
- [14] Miyazono, Koki; Kagarise, Christopher D.; Koelling, Kurt W.; Mahboob, Monon; Bechtel, Stephen E. Shear and Extensional Rheology and Flow-Induced Orientation of Carbon Nanofiber/Polystyrene Melt Composites. *Journal of Applied Polymer Science* **2011**, Vol. 119, 1940–1951.
- [15] Glasgow, D.G.; Jacobsen, R.L.; Burton, D.J.; Kwag, C.; Kennel, E.; Lake, M.L.; Brittain, W.J.; Rice, B.P. Carbon nanofiber polymer composites. *International*

- SAMPE symposium and exhibition **2003**, 48(advancing materials in the global economy—applications, emerging markets and evolving technologies, Book 2)
- [16] Lake, M.L.; Glasgow, D.G.; Kwag, C.; Burton, D.J. Carbon nanofiber polymer composites: electrical and mechanical properties. Int SAMPE Symp **2002**, Exhibit 47:1794–1800
- [17] Shi, D.; Lian, J.; He, P.; Wang, L.M.; Xiao, F.; Yang, L.; Schulz, M.J.; Mast, D.B. Plasma coating of carbon nanofibers for enhanced dispersion and interfacial bonding in polymer composites. Appl Phys Lett **2003**, 83(25):5301–5303
- [18] Enomoto, K.; Yasuhara, T.; Kitakata, S.; Murakami, H.; Ohtake, N. Frictional properties of carbon nanofiber reinforced polymer matrix composites. New Diamond Frontier Carbon Technol **2004**, 4(1):11–20
- [19] McKenzie, J.L.; Waid, M.C.; Shi, R.; Webster, T.J. Decreased functions of astrocytes on carbon nanofiber materials. Biomaterials **2003**, 25(7–8):1309–1317
- [20] Lozano, K.; Yang, S.; Zeng, Q. Rheological analysis of vapor-grown carbon-nanofiber reinforced polyethylene composites. J. Appl. Polym. Sci. **2004**, 93, 155-162.
- [21] Hammel, E.; Tang, X.; Trampert, M.; Schmitt, T.; Mauthner, K.; Eder, A.; Pötschke, P. Carbon nanofibers for composite applications. Carbon **2004**, 42, 1153-1158.
- [22] Tibbetts, G. G.; McHugh, J. J. Mechanical properties of vapor-grown carbon fiber composite with thermoplastic matrices. J. Mater. Res. **1999**, 14, 2871-2880.
- [23] Van Hattum, F. W. J.; Bernardo, C. A.; Finegan, J. C.; Tibbetts, G. G.; Alig, R. L.; Lake, M. L. A study of the thermomechanical properties of carbon fiber composites. Polym. Compos. **1999**, 20, 683-688.

- [24] Carneiro, O. S.; Maia, J. M. Polym Compos **2000**, 21, 960.
- [25] Lozano, K.; Barrera, E. V. Nanofiber-reinforced thermoplastic composites: I. Thermoanalytical and mechanical analyses. J. Appl. Polym. Sci. **2001**, 79, 125-133.
- [26] Lozano, K.; Bonilla-Rios, J.; Barrera, E. V. A study on nanofiber-reinforced thermoplastic composites: II Investigation the mixing rheology and conduction properties. J. Appl. Polym. Sci. **2001**, 80, 1162-1172.
- [27] Ceccia, S.; Ferri, D.; Tabuani, D.; Maffettone, P. L. Rheology of carbon nanofiber-reinforced polypropylene. Rheol. Acta. **2008**, 47(4), 425-433.
- [28] Kumar, S.; Doshi, H.; Srinivasarao, M.; Park, J. O.; Schiraldi, D. A. Fibers from polypropylene/nano carbon fiber composites. Polymer **2002**, 43, 1701-1703.
- [29] Carneiro, O. S.; Maia, J. M. Rheological behavior of (short) carbon fiber/thermoplastic composites Part II: the influence of matrix type. Polymer Composites **2000**, 21, 970-977.
- [30] Caldeira, G.; Maia, J. M.; Carneiro, O. S.; Covas, J. A.; Bernardo, C. A. Production and characterization of innovative carbon fiber polycarbonate composites. Polymer Composites **1998**, 19, 147-151.
- [31] Carneiro, O. S.; Covas, J. A.; Bernardo, C. A.; Caldeira, G.; Van Hattum, F. W. J.; Ting, J.-M.; Alig, R. L.; Lake, M. L. Production and assessment of polycarbonate composites reinforced with vapour-grown carbon fibres. Compos. Sci. Technol. **1998**, 58, 401-407.
- [32] Higgins, B. A.; Brittain, W. J. Polycarbonate carbon nanofiber composites. Eur. Polym. J. **2005**, 41, 889-893.

- [33] Ma, H.; Zeng, J.; Realff, M. L.; Kumar, S.; Schiraldi, D. A. Processing, structure, and properties of fibers from polyester/carbon nanofiber composites. *Compos. Sci. Technol.* **2003**, 63, 1617-1628.
- [34] Tibbetts, G. G.; McHugh, J. J. Mechanical properties of vapor-grown carbon fiber composites with thermoplastic matrices. *J. Mater. Res.* **1999**, 14, 2871-2880.
- [35] Lake, M. L.; Glasgow, D. G.; Kwag, C.; Burton, D. J. In Proceedings of 47th International SAMPE Symposium and Exhibition; Society for the Advancement of Material and Process Engineering, Covina, California, 2002; p 1794.
- [36] Cooper, C. A.; Ravich, D.; Lips, D.; Mayer, J.; Wagner, H. D. Distribution and alignment of carbon nanotubes and nanofibrils in a polymer matrix. *Compos. Sci. Technol.* **2002**, 62, 1105-1112.
- [37] Zeng, J.; Saltysiak, B.; Johnson, W. S.; Schiraldi, D. A.; Kumar, S. Processing and properties of poly(methyl methacrylate)/carbon nanofiber composites. *Composites Part B* **2004**, 35, 173-178.
- [38] Cipriano, B. H.; Kota, A. K.; Gershon, A. L.; Laskowski, C. J.; Kashiwagi, T.; Bruck, H. A.; Raghavan, S. R. Conductivity enhancement of carbon nanotube and nanofiber-based polymer nanocomposites by melt annealing. *Polymer* **2008**, 49, 4846-4851.
- [39] Murch, William L. The effect of flow-induced orientation on the rheology of carbon nanofiber/polystyrene composites during flow reversal experiments. Undergraduate Thesis, The Ohio State University Knowledge Bank, **2011**.

- [40] Kremer, Timothy. Improvement of a constitutive model for predicting flow behavior of nanocomposites. Undergraduate Thesis, The Ohio State University Knowledge Bank, **2013**.
- [41] Miyazono, K.; Kagarise, C.D.; Mahboob, M.; Bechtel, S.E.; Koelling, K.W. The effect of length, dispersion and surface treatment on the rheological behavior of carbon nanofiber/polystyrene melt composites. (manuscript in preparation)
- [42] Pyrograf Products, Inc. Pyrograf-III carbon nanofiber.
http://pyrografproducts.com/Merchant5/merchant.mvc?Screen=cp_nanofiber
 (accessed Apr. 5, 2015).
- [43] Mahboob, M.; Kagarise, C.; Koelling, K.W.; Bechtel, S.E. (in press) Quantitative 3D measurement of the nanostructural features that dictate mesoscale performance properties of nanocomposites. *Polym Compos*
- [44] Azaiez, Jalel. Constitutive equations for fiber suspensions in viscoelastic media. *J. Non-Newtonian Fluid Mech.* **1996**, 66, 35-54.
- [45] Bird, R.B., Curtiss, C.F., Armstrong, R.C., and Hassager, O. (1987) Dynamics of polymeric liquids. Wiley, New York.
- [46] Tucker, C.L. Flow regimes for fiber suspensions in narrow gaps. *J Non-Newton Fluid Mech* **1991**, 39(3): 239-268.
- [47] Enomoto, K.; Yasuhara, T.; Kitakata, S.; Murakami, H.; Ohtake, N. Frictional properties of carbon nanofiber reinforced polymer matrix composites. *New Diamond Frontier Carbon Technol.* **2004**, 14, 11-20.

- [48] Shofner, M.L.; Lozano, K.; Rodriguez-Macias, F.J.; Barrera, E.V.. Nanofiber-reinforced polymers prepared by fused deposition modeling. *J. Appl. Polym. Sci.* **2003**, 89, 3081-3090.
- [49] Pogue, R.T.; Ye, J.; Klosterman, D.A.; Glass, A.S.; Chartoff, R.P. Evaluating fiber-matrix interaction in polymer-matrix composites by inverse gas chromatography. *Compos. A-Appl. Sci. Manuf.* **1998**, 29, 1273-1281.
- [50] Advani, S.G.; Tucker, C.L. The use of tensors to describe and predict fiber orientation in short fiber composites. *J. Rheol.* **1987**, 31, 751-784.
- [51] Mahboob,M; Kagarise,C; Koelling,K,W; Bechtel,S,E. Quantitative 3D Measurement of the Nanostructural Features that Dictate Mesoscale Performance Properties of Nanocomposites. *Polymer Composites* **2010**, 31, no. 9, 1495 - 1503.
- [52] Advani, S. G.; Tucker, C. L. I. Closure approximations for three-dimensional structure tensors. *J. Rheol.***1990**, 34, 367–386
- [53] Folgar, F.; Tucker, C.L. Orientation behaviour of fibres in concentrated suspensions. *J Reinf Plast Compos* **1984**, 3, 98-119.
- [54] The MathWorks Inc. Choosing a Solver.
<http://www.mathworks.com/help/optim/ug/choosing-a-solver.html#brppuoz>
(accessed Apr 8, 2015)

Appendix A: Programs

Constitutive Model Solver: Extensional Flow	56
Constitutive Model Solver: Shear Flow	59
Pure Polymer: Overhead Parameter Optimization	62
Pure Polymer: SAOS Flow	65
Pure Polymer: Transient Shear Flow	67
Pure Polymer: Transient Extensional Flow	72
Determining Boundary of Convergence for σ , λ , and α	75
Composite Parameter Optimization: C_1	82
Composite Parameter Optimization: Aspect Ratio Scaling Factor	84
Composite Parameter Optimization: σ	91
Composite Model Predictions: Melt Blended with O-CNFS (MB)	94
Composite Model Predictions: Melt Blended with HHT-CNFs (MBHHT)	103
Composite Model Predictions: Solvent Cast with O-CNFs (SC)	111
Composite Model Predictions: Fitting G' and G''	119
Composite Model Predictions: Solving the Constitutive Model for SAOS Flow	121
Composite Model Predictions: Modeling the Stress Wave in SAOS Flow	125

Constitutive Model Solver: Extensional Flow

```
function [time, etac, diverge] =  
extensional_pde_solver(r,mass,sigma,CI,etap,lambda,alpha,tspan,modes,re  
,orientation)  
% This program solves the constitutive model equations for extensional  
flow  
  
%dummy variables used to catch divergence  
diverge=0; p=0;  
total12=0;  
  
chi=1.0*(re^2-1)/(re^2+1); %another form of aspect ratio  
rf=1750.0; %fiber density  
rs=1000.0; %polymer density  
  
phi=rs*mass/(rf+(rs-rf)*mass);%volume fract of CNFs  
Ap=re^2/(3*log(sqrt(pi/phi))); %shape factor, aka A_2  
  
initial=[0 0 0 orientation]; %initial conditions for the differential  
equation solver  
  
for x=1:modes  
    %solves differential equations  
    [time, yout]=ode23tb(@modeextensionalsub,tspan,initial);  
  
    tau11=yout(:,1);  
    tau22=yout(:,2);  
    tau12=(tau11-tau22); %not actually tau12, just the difference  
    between 11 and 22  
    length_tau=length(tau12); %dummy variable for divergence  
  
    if p==0 %in first loop to set up the length of tau when solution  
    converges  
        total12=total12+tau12;  
        converge_length=length(total12);  
        if mass~=0  
            a11=yout(:,4);  
            a22=yout(:,5);  
            a33=yout(:,6);  
            a11out(:,1)=a11;  
            a22out(:,1)=a22;  
            a33out(:,1)=a33;  
            timeout(:,1)=time;  
        end  
        p=p+1;  
    elseif p>0 %after the first loop, this checks to see if tau's are  
    all same length as previous mode  
        if length_tau<converge_length %if not, then the solution  
        diverged  
            diverge=1;  
            break %get out of this program because current conditions  
            cause divergence  
        end
```

```

total12=total12+tau12;
converge_length=length(total12);

if mass~=0
    a11=yout(:,4);
    a22=yout(:,5);
    a33=yout(:,6);
    allout(:,x)=a11;
    a22out(:,x)=a22;
    a33out(:,x)=a33;
    timeout(:,x)=time;
end
end
end

eta=total12./r; %definition of extensional viscosity
if mass==0
    etac=eta; %viscosity of pure polymer = the viscosity from the
polymer
else
    a11_out=allout(:,1); %the 11 component of orientation
    %stress due to cnfs
    coef=2.0.*eta*phi;
    tf12=(coef).*(Ap*r*((27.0*(a11.*a22.*a33)).*(-
3/35+(4*a11+2*a22)/7)+(1-27*(a11.*a22.*a33)).*(a11.^2+(-3.*a11.*a22-
a11.*a33+a22.^2+a22.*a33)/2)));
    etac=tf12/r+eta; %viscosity of a composite = the viscosity from the
polymer + stress/rate from the cnfs
end

function dy = modeextensionalsub(t,y)
% this sub-function includes the differential equations for polymer
stress
% and cnf orientation evolution

    %stress tensor components
    t11=y(1);
    t22=y(2);
    t33=y(3);
    %orientation tensor components
    a11=y(4);
    a22=y(5);
    a33=y(6);

    %polymer stress differential equations
    dy=zeros(6,1);
    dy(1,1)=2*r*etap(x)/lambda(x)-sigma*t11/lambda(x)-
alpha(x)/etap(x)*t11^2+2*r*t11-3*(1-sigma)*2*a11*t11/lambda(x);
    dy(2,1)=-r*etap(x)/lambda(x)-sigma*t22/lambda(x)-
alpha(x)/etap(x)*t22^2-r*t22-3*(1-sigma)*a22*t22/lambda(x);
    dy(3,1)=-r*etap(x)/lambda(x)-sigma*t33/lambda(x)-
alpha(x)/etap(x)*t33^2-r*t33-3*(1-sigma)*a33*t33/lambda(x);

```

```

    if mass~=0
        %cnf orientation evolution equations
        dy(4,1)=CI*2*3^(.5)*r*(1-3*a11)+2*chi*r*a11-
        2*chi*r*(27*a11*a22*a33*(-2/35+(10*a11-a22-a33)/14)+(1-
        27*a11*a22*a33)*(a11^2-(a11*a22+a11*a33)/2));
        dy(5,1)=CI*2*3^(.5)*r*(1-3*a22)-chi*r*a22-
        2*chi*r*(27*a11*a22*a33*(1/35+(2*a11-5*a22-a33)/14)+(1-
        27*a11*a22*a33)*(a11*a22-(a22^2+a22*a33)/2));
        dy(6,1)=CI*2*3^(.5)*r*(1-3*a33)-chi*r*a33-
        2*chi*r*(27*a11*a22*a33*(1/35+(2*a11-a22-5*a33)/14)+(1-
        27*a11*a22*a33)*(a11*a33-(a22*a33+a33^2)/2));
    end
end

end

```

Constitutive Model Solver: Shear Flow

```
function [time1, etacf,diverge] =  
shear_pde_solver(r,mass,sigma,CI,etap,lambda,alpha,tspan,modes,re,orien  
tation)  
% This program solves the constitutive model equations for shear flow  
  
%dummy variables used to catch divergence  
diverge=0; p=0;  
total12=0;  
  
chi=1.0*(re^2-1)/(re^2+1); %another form of aspect ratio  
rf=1750.0; %fiber density  
rs=1000.0; %polymer density  
phi=rs*mass/(rf+(rs-rf)*mass); %volume fract of CNFs  
Ap=re^2/(3*log(sqrt(pi/phi))); %shape factor, aka A_2  
  
initial=[0 0 0 0 orientation]; %initial conditions for the differential  
equation solver  
  
tau_finalf=zeros(4,modes);  
  
for x=1:modes  
    [time1, yo]=ode23tb(@modeshearsub,tspan,initial); %ode solver  
    for solving Equations (2) & (4) simultaneously  
  
        length_tau=length(yo(:,1)); %dummy variable to catch divergence  
        L=length(time1);  
  
        if x==1  
            tau1=zeros(L,modes);  
            tau12=zeros(L,modes);  
            tau2=zeros(L,modes);  
            tau3=zeros(L,modes);  
        end  
        if p==0 %in first loop to set up the length of tau when solution  
converges  
            tau1(:,x)=yo(:,1);  
            converge_length=length(tau1);  
            tau12(:,x)=yo(:,2);  
            tau2(:,x)=yo(:,3);  
            tau3(:,x)=yo(:,4);  
            tau_finalf(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x) tau3(L,x)]';  
  
            p=p+1;  
        elseif p>0 %after the first loop, this checks to see if tau's are  
all same length as previous mode  
            if length_tau<converge_length %if not, then the solution  
diverged  
                diverge=1;  
                break %get out of this program because current conditions  
cause divergence  
            end  
            tau1(:,x)=yo(:,1);
```

```

        converge_length=length(tau1);
        tau12(:,x)=yo(:,2);
        tau2(:,x)=yo(:,3);
        tau3(:,x)=yo(:,4);

        tau_finalf(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x) tau3(L,x)]';
    end
end

if diverge==0
total11f=sum(tau1,2);
total12f=sum(tau12,2);
total22f=sum(tau2,2);
total33f=sum(tau3,2);

if mass~=0
    a11f=yo(:,5);
    a12f=yo(:,6);
    a22f=yo(:,7);
    a33f=1-a11f-a22f;
    a_final = [a11f(L) a12f(L) a22f(L)]';
    a11final_shr=a11f(end);
end
T_final= [total11f(L) total12f(L) total22f(L) total33f(L)];

etaf=total12f./r; %definition of shear viscosity
coef=2.0*etaf*phi;

%%%This section computes the fiber stress from Equation (3)
if mass==0
    etacf=etaf; %viscosity of pure polymer = the viscosity from the
polymer
else
    %stress due to cnfs
    tf12f=(2.0*phi.*total12f.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-
(a12f.^2).*(1-a11f-a22f))).*(-
1.0/35.0+1.0/7.0*(a11f+a22f)))+(a12f.^2).*(1-27*(a11f.*a22f.*(1-a11f-
a22f))+27.0*(a12f.^2).*(1-a11f-a22f)));
    tf11f=(2.0*phi.*total12f.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-
(a12f.^2).*(1-a11f-a22f))).*(3.0/7.0*(a12f)))+(a11f.*a12f).*(1-
27*(a11f.*a22f.*(1-a11f-a22f))+27.0*(a12f.^2).*(1-a11f-a22f)));
    tf22f=(2.0*phi.*total12f.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-
(a12f.^2).*(1-a11f-a22f))).*(3.0/7.0*(a12f)))+(a22f.*a12f).*(1-
27*(a11f.*a22f.*(1-a11f-a22f))+27.0*(a12f.^2).*(1-a11f-a22f)));
    tf33f=(2.0*phi.*total12f.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-
(a12f.^2).*(1-a11f-a22f))).*(1.0/7.0*(a12f)))+(1-a11f-a22f).*(a12f).*(1-
27*(a11f.*a22f.*(1-a11f-a22f))+27.0*(a12f.^2).*(1-a11f-a22f)));

    tauc12f=tf12f+total12f;
    tauc11f=tf11f+total11f;
    tauc22f=tf22f+total22f;
    tauc33f=tf33f+total33f;
    etacf=tauc12f./r; %viscosity of composite = the viscosity from the
polymer + stress/rate from the cnfs

```

```

end

elseif diverge==1
    etacf=0;
end

function dyo = modeshearsub(t,y)
% this sub-function includes the differential equations for polymer
stress
% and cnf orientation evolution

    %stress tensor components
    tp11=y(1);
    tp12=y(2);
    tp22=y(3);
    tp33=y(4);
    %orientation tensor components
    a11=y(5);
    a12=y(6);
    a22=y(7);
    a33=1-a11-a22;
    dyo=zeros(7,1);

    %polymer stress differential equations
    dyo(1,1)=-alpha(x)*(tp11^2+tp12^2)/etap(x)-
sigma*tp11/lambda(x)+2*r*tp12-3*(1-
sigma)*(tp11*a11+tp12*a12)/lambda(x);
    dyo(2,1)=-alpha(x)*(tp11*tp12+tp12*tp22)/etap(x)-
sigma*tp12/lambda(x)+etap(x)*r/lambda(x)+r*tp22-3*(1-
sigma)/2/lambda(x)*(a11*tp12+a12*tp22+a12*tp11+a22*tp12);
    dyo(3,1)=-alpha(x)*(tp12^2+tp22^2)/etap(x)-sigma*tp22/lambda(x)-
3*(1-sigma)/lambda(x)*(tp12*a12+tp22*a22);
    dyo(4,1)=-alpha(x)*(tp33^2)/etap(x)-tp33*sigma/lambda(x)-3*(1-
sigma)/lambda(x)*a33*tp33;

    if mass~=0
        %cnf orientation evolution equations
        dyo(5,1)=r*a12+2*CI*abs(r)*(1.0-3.0*a11)+chi*r*a12-...
2*chi*r*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)...
-27.0*(a12^2)*(1-a11-a22)))+(1.0-27.0*a11*a22*(1-a11-
a22))+...
27.0*(a12^2)*(1-a11-a22))*a11*a12);
        dyo(6,1)=1.0/2.0*r*a22-1.0/2.0*r*a11+chi*((1.0/2.0*r*a22+...
1.0/2.0*r*a11)-(2.0*r)*((27.0*a11*a22*(1-a11-a22)-...
27.0*(a12^2)*(1-a11-a22))*(-
1.0/35.0+1.0/7.0*a11+1.0/7.0*a22))+...
(1.0-27.0*a11*a22*(1-a11-a22)...
+27.0*(a12^2)*(1-a11-a22))*a12^2))-6.0*CI*abs(r)*a12;
        dyo(7,1)=-r*a12+2*CI*abs(r)*(1.0-3.0*a22)+chi*r*a12-...
2*chi*r*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)-...
27.0*(a12^2)*(1-a11-a22)))+(1.0-27.0*a11*a22*(1-a11-a22))+...
27.0*(a12^2)*(1-a11-a22))*a12*a22);
    end
end

end

```

Pure Polymer: Overhead Parameter Optimization

```
function [] = Pure_Polymer_Optimization()
% This program optimizes the pure polymer parameters (etap, lambda, and
% alpha) using transient extension, transient shear, and SAOS data.
% This program also plots model predictions for all three flows

format long g
z=.01; %relative error scaling factor, number to divide Gerror by
y=10; %relative error scaling factor, number to divide Serror by

modes=6; %number of modes for Giesekus model
solved=1; % =0 optimize (use fmincon), =1 already solved (don't use
fmincon)

%initial guesses for pure polymer parameters in fmincon: etap, lambda,
and alpha
Eta0=[4.068958213059190e+4    0.161814434808827e+4
0.926492125161873e+4    0.889129038018490e+4    3.023078079743952e+4
0.824206633606583e+4];
lambda=[1.162556126233308e+5    0.000000152130995e+5
0.064530255537243e+5    0.000002089102109e+5    0.000026539367359e+5
0.000302338431186e+5];
alpha=[0.001745546564408    0.851749611132819    0.027375867196483
0.799680223515555    0.471561581048678    0.072223157940858];

% Final results:
% if previously optimized, plug in results here for quick plotting
Final=[    1563.180023 10503.23358 30793.85272 9864.433234 9310.030642
40690.01669; %eta by mode
        0.01461114 0.235979496 3.210427777 34.39722848 6390.005373
116255.5053; %lambda by mode
        0.818868357 0.56          0.56          0.078601046 0.025189799
0.001841609]; %alpha by mode

if solved==0
    options=optimset('Algorithm','interior-point'); %search
algorithm
    f0=[Eta0; lambda; alpha]; %initial guesses for fmincon

    % Note on fmincon: the two matrices passed to fmincon are the
lower
    % bounds and upper bounds on the parameters
    % (column = mode, row = eta; lambda; alpha)
    [Final, fval, exitflag]=fmincon(@FIT_0wt,f0,[],[],[],[],[ 0 0 0
0 0 0; 0 0 0 0 0 0; 0 0 0 0 0 0],[inf inf inf inf inf inf; inf inf inf
inf inf inf; 0.999 0.999 0.999 0.999 0.999 0.999],[],options);

    % Adjust Final lambda and alpha values to converging region
    % Note: this is actually what fmincon saw for values of lambdas
and
    % alphas since it was changed within its operation, but the
values
    % it returns may not be the ones it used for error
minimization,
```

```

% hence this for-loop is needed to change the values back to
what
% was used in the error minimization
for q=1:modes %these convergences are based on sigma=0.5,
CI=.01
    if Final(2,q)>0.07 && Final(2,q)<=0.11 && Final(3,q)>0.7
        Final(3,q)=0.7;
    elseif Final(2,q)>0.11 && Final(2,q)<=0.18 &&
Final(3,q)>0.6
        Final(3,q)=0.6;
    elseif Final(2,q)>0.18 && Final(2,q)<=0.27 &&
Final(3,q)>0.56
        Final(3,q)=0.56;
    elseif Final(2,q)>0.27 && Final(2,q)<=0.94 &&
Final(3,q)>0.52
        Final(3,q)=0.52;
    elseif Final(2,q)>0.94 && Final(2,q)<=1.43 &&
Final(3,q)>0.54
        Final(3,q)=0.54;
    elseif Final(2,q)>1.43 && Final(2,q)<=465 &&
Final(3,q)>0.56
        Final(3,q)=0.56;
    elseif Final(2,q)>465 && Final(2,q)<=1061 &&
Final(3,q)>0.58
        Final(3,q)=0.58;
    elseif Final(2,q)>1061 && Final(2,q)<=1604 &&
Final(3,q)>0.6
        Final(3,q)=0.6;
    elseif Final(2,q)>1604 && Final(2,q)<=2425 &&
Final(3,q)>0.64
        Final(3,q)=0.64;
    elseif Final(2,q)>2425 && Final(2,q)<=3666 &&
Final(3,q)>0.74
        Final(3,q)=0.74;
    end
end
end

% Plot model vs experiment
graph=1;
figure
[GerrorFinal]=SAOS_MB_0wt(Final,graph,modes)
figure
[TerrorFinal]=Extensional_MB_0wt(Final,graph,modes)
figure
[SerrorFinal]=Shear_MB_0wt(Final,graph,modes)

% Display final errors and values
GerrorFinal/z
TerrorFinal
SerrorFinal/y
e=TerrorFinal+GerrorFinal+SerrorFinal

etafinal=Final(1,:) '
lambdafinal=Final(2,:) '
alphafinal=Final(3,:) '

```



```

function e = FIT_0wt(para)
%this subfunction calculates the model predictions for all three flow
%fields and the error between the model predictions and experiment
    graph=0;
    Eta=para(1,:);
    Lambda=para(2,:);
    Alpha=para(3,:);

    % This for-loop constricts lambda and alpha values to the
    converging region previously determined by "Parameter_Convergence"
    program.
    % Note: this region is dependent on sigma, so if you want to
    examine lower values of sigma in future composite fittings, you need to
    refit the converging region, alter this for-loop to constrict lambdas
    and
    % alphas, and refit the pure polymer.
    for g=1:modes %these convergences are based on sigma=0.5, CI=.01
        if Lambda(g)>0.07 && Lambda(g)<=0.11 && Alpha(g)>0.7
            Alpha(g)=0.7;
        elseif Lambda(g)>0.11 && Lambda(g)<=0.18 && Alpha(g)>0.6
            Alpha(g)=0.6;
        elseif Lambda(g)>0.18 && Lambda(g)<=0.27 && Alpha(g)>0.56
            Alpha(g)=0.56;
        elseif Lambda(g)>0.27 && Lambda(g)<=0.94 && Alpha(g)>0.52
            Alpha(g)=0.52;
        elseif Lambda(g)>0.94 && Lambda(g)<=1.43 && Alpha(g)>0.54
            Alpha(g)=0.54;
        elseif Lambda(g)>1.43 && Lambda(g)<=465 && Alpha(g)>0.56
            Alpha(g)=0.56;
        elseif Lambda(g)>465 && Lambda(g)<=1061 && Alpha(g)>0.58
            Alpha(g)=0.58;
        elseif Lambda(g)>1061 && Lambda(g)<=1604 && Alpha(g)>0.6
            Alpha(g)=0.6;
        elseif Lambda(g)>1604 && Lambda(g)<=2425 && Alpha(g)>0.64
            Alpha(g)=0.64;
        elseif Lambda(g)>2425 && Lambda(g)<=3666 && Alpha(g)>0.74
            Alpha(g)=0.74;
        end
    end

    para1=[Eta;Lambda;Alpha]; %new 'para' with adjusted lambdas and
    alphas

    [Gerror]=SAOS_MB_0wt(para1,graph,modes);
    [Terror]=Extensional_MB_0wt(para1,graph,modes);
    [Serror]=Shear_MB_0wt(para1,graph,modes);

    Gerror=Gerror/z;
    Serror=Serror/y;
    e=Terror+Gerror+Serror;

end
end

```

Pure Polymer: SAOS Flow

```
function [Gerror] = SAOS_MB_0wt(para,graph,modes)
%This program calculates the error between this model prediction of
moduli
%vs frequency and the experimental data and sends it back to
%Pure_Polymer_Optimization to be used in the error minimization for
%parameter optimization. This program can also plot the model
%prediction of moduli given the set of pure polymer parameters

eta=para(1,:);
lambda=para(2,:);

%from excel: 0wt%SAOS and Model fitting: Fit to extension
freq=[100 63.096 39.811 25.119 15.849 10 6.3096 3.9811 2.5119
1.5849 1 0.63096 0.39811 0.25119 0.15849 0.1 0.063096 0.039811
0.025119 0.015849 0.01];
%storage modulus
Gexp=[1.15E+05 99721 85344 72157 59959 48938 39138 30504
23217 17157 12214 8405.3 5602.5 3536.2 2187.2 1252.3 701.47
378.3 185.27 93.604 42.558];
%loss modulus
GDexp=[57682 51912 47430 43100 38967 34927 30883 26877
22964 19168 15682 12477 9636 7275.6 5317.9 3783.3 2614.8
1782.6 1178.2 766.23 482.01];

%graph G'&G''
Gerror=0;
Gprime=zeros(length(freq),1);
GDprime=zeros(length(freq),1);
Gerror=0;
    %G' & G'' calculation
    for j=1:length(freq)
        Gtemp=zeros(1,5);
        GDtemp=zeros(1,5);
        for i=1:modes
            Gtemp(i)=eta(i)*lambda(i)*freq(j)^2/(1+(lambda(i)*freq(j))^2);
            GDtemp(i)=eta(i)*freq(j)/(1+(lambda(i)*freq(j))^2);
        end

        %calculate G' and G'' as sum of modes
        Gprime(j)=sum(Gtemp);
        GDprime(j)=sum(GDtemp);
        %calculate error between model and experiment
        Gerror=Gerror+(log10(Gexp(j))-
log10(Gprime(j)))^2+(log10(GDexp(j))-log10(GDprime(j)))^2;
    end
%plot the model predictions and experimental data
if graph==1

loglog(freq,Gprime,'b',freq,Gexp,'+b',freq,GDprime,'g',freq,GDexp,'^g')
;
    title('G-Prime and GDouble-Prime MB 0wt%');
    xlabel('Frequency (rad/s)');
```

```
ylabel('Gprime,GDprime (Pa*s)');  
legend('Model Gprime','Exp Gprime','Model GDprime','Exp GDprime',-  
1);  
end  
end
```

Pure Polymer: Transient Shear Flow

```
function [Error] = Shear_MB_0wt(para,graph,modes)
%This program calls on shear_pde_solver to solve the transient
%differtial equations from the constitutive model and create a model
%prediction for the transient shear viscosity vs time. The error
%between this model prediction and the experimental data is calculated
and
%sent back to Pure_Polymer_Optimization to be used in the error
minimization
%for parameter optimization. This program can also plot the model
%prediction of transient shear viscosity given the set of pure polymer
%parameters

sigma=1; %no cnfs - value doesn't matter
CI=0; %no cnfs - value doesn't matter
wt=0; %no cnfs - value doesn't matter
re=0; %no cnfs - value doesn't matter
orient=[0 0 0]; %no cnfs - value doesn't matter

Error=zeros(1,5); %initialize error variable

for w=1:5 % w designates the w-th shear rate
    if w==1
        r=.01;
        %ave data - can truncate at t= 95.28, exp= 47586
        timedata=[0 0.02 0.03 0.04 0.05 0.06 0.07
0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15 0.16
0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.24 0.25
0.26 0.27 0.28 0.29 0.3 0.31 0.32 0.33 0.34
0.35 0.36 0.37 0.38 0.39 0.4 0.41 0.42 0.43
0.44 0.45 0.46 0.47 0.48 0.49 0.5 0.51 0.52
0.53 0.54 0.55 0.56 0.57 0.58 0.59 0.6 0.61
0.62 0.63 0.64 0.65 0.66 0.685 0.725 0.77 0.82
0.875 0.935 0.995 1.06 1.13 1.205 1.285 1.37 1.465
1.565 1.665 1.775 1.895 2.02 2.155 2.295 2.445 2.61
2.78 2.96 3.16 3.37 3.59 3.83 4.085 4.355 4.645
4.955 5.285 5.635 6.01 6.41 6.835 7.285 7.765 8.285
8.835 9.42 10.045 10.71 11.42 12.18 12.99 13.85 14.77
15.75 16.795 17.91 19.1 20.37 21.72 23.16 24.7 26.34
28.09 29.955 31.945 34.065 36.325 38.74 41.31 44.05 46.975
50.095 53.425 56.975 60.755 64.785 69.085 73.675 78.57 83.785
89.345 95.28 101.61 108.35 115.55 123.22 131.4 140.13 149.43
159.35 169.93 181.22 193.25 206.08 219.77 234.36 249.92 266.52
284.22 303.1 333.6 355.76 379.38 404.57 431.43 460.07 490.63
523.21 557.96 594.99 634.51 676.64 721.58 769.48 820.58 875.07
933.17 995.13 1061.2 1131.7 1206.8 1287 1372.4 1463.6 1560.7
1664.4 1774.9 1892.8 2018.4 2152.5 2295.4 2447.8 2610.4 2783.7
2968.5 3165.7 3375.9];
        expdata=[0 217 945.14 1718.2 2445.9 3001.6 3530.6
4035.4 4727 5198.9 5287.9 5707.5 6336.1 6785.1 6768.4 7095.1
7532.2 7921.6 8225.2 8373.8 8640 8774.3 9115.2 9558.7 9609.5
9786 10121 10407 10693 10571 10861 11203 11381 11647
11717 11869 12065 12414 12606 12594 12751 13024 13142
13288 13449 13671 13740 13927 14033 14050 14208 14414
```

```

14538    14678    14871    15031    15006    15119    15238    15337    15505
15823    15757    15984    16090    16315    16476    16870    17342    17833
18359    18908    19428    20005    20435    20954    21541    22141    22752
23371    23920    24414    25054    25631    26223    26748    27206    27877
28562    29154    29682    30316    30947    31437    32022    32633    33239
33829    34347    34913    35470    36032    36589    37054    37442    37888
38263    38697    39120    39486    39934    40320    40757    41061    41453
41795    42157    42737    43150    43440    43635    43756    43833    43980
44368    44750    44994    45227    45406    45355    45513    45978    46470
46466    46549    46714    46909    46765    47036    47159    46912    47238
47702    47586    47660    47749    47735    47748    48108    47970    48142
47937    48534    48367    48457    48642    48708    48621    48914    48943
48970    49099    49102    49219    49213    49142    49399    49436    49473
49477    49630    49629    49634    49657    49467    49573    49532    49449
49332    49334    49204    49068    48875    48756    48780    48845    48595
48665    48405    48407    48195    48307    48154    47951    47885    47777
47499    47408    47395];
elseif w==2
    r=.1;
    %ave data - can truncate at t= 95.28, exp= 41663.66667
    timedata=[0 0.01    0.02    0.03    0.04    0.05    0.06
0.07    0.08    0.09    0.1 0.11    0.12    0.13    0.14    0.15
0.16    0.17    0.18    0.19    0.2 0.21    0.22    0.23    0.24
0.25    0.26    0.27    0.28    0.29    0.3 0.31    0.32    0.33
0.34    0.35    0.36    0.37    0.38    0.39    0.4 0.41    0.42
0.43    0.44    0.45    0.46    0.47    0.48    0.49    0.5 0.51
0.52    0.53    0.54    0.55    0.56    0.57    0.58    0.59    0.6
0.61    0.62    0.63    0.64    0.65    0.66    0.685    0.725    0.77
0.82    0.875    0.935    0.995    1.06    1.13    1.205    1.285    1.37
1.465    1.565    1.665    1.775    1.895    2.02    2.155    2.295    2.445
2.61    2.78    2.96    3.16    3.37    3.59    3.83    4.085    4.355
4.645    4.955    5.285    5.635    6.01    6.41    6.835    7.285    7.765
8.285    8.835    9.42    10.045    10.71    11.42    12.18    12.99    13.85
14.77    15.75    16.795    17.91    19.1    20.37    21.72    23.16    24.7
26.34    28.09    29.955    31.945    34.065    36.325    38.74    41.31    44.05
46.975    50.095    53.425    56.975    60.755    64.785    69.085    73.675    78.57
83.785    89.345    95.28    101.61    108.35    115.55    123.22    131.4    140.13
149.43    159.35    169.93    181.22    193.25    206.08    219.77    234.36    249.92
266.52    284.22    303.1    333.6    355.76    379.38    404.57    431.43    460.07
490.63    523.21    557.96    594.99    634.51    676.64    721.58    769.48    820.58
875.07    933.17    995.13    1061.2    1131.7    1206.8    1287    1372.4    1463.6
1560.7    1664.4    1774.9    1892.8    2018.4    2152.5    2295.4    2447.8    2610.4
2783.7    2968.5    3165.7    3375.9];
    expdata=[0 44.674    445.1933333    1038.633333    1678.366667
2264.7    2822.4    3330.333333    3806.566667    4234.2    4643.9    5025.1
5395.533333    5726.833333    6028.733333    6346.066667    6652.933333    6924.633333
7195.466667    7477.433333    7727.4    7978.633333    8221.6    8470.866667    8676.4
8900.6    9136.5    9330.433333    9541.4    9716.4    9923.1    10113.76667
10317.16667    10496.36667    10680.86667    10862    11040    11216.66667    11393
11565    11712.33333    11884.33333    12023.66667    12214.66667    12363
12493.66667    12650    12786    12937    13069.66667    13222.33333    13383.33333
13531.33333    13683    13810.66667    13937.66667    14077    14205.66667    14332
14443.66667    14581.66667    14684.66667    14806    14931.66667    15024.66667
15102    15272.66667    15567    16007    16492.33333    16973.66667    17496.33333
18016.66667    18500    18962.33333    19504    19993    20538.33333    21050
21640    22208.66667    22761.33333    23311.33333    23908.66667    24509.33333
25044    25588    26149.66667    26691    27190    27764.33333    28404.66667

```

```

28996.33333 29527 30056.33333 30566.66667 31120 31719 32299.66667
32764.33333 33265 33856.33333 34346.66667 34773 35272.33333
35753.66667 36167.66667 36639.33333 37008.33333 37470.66667 37817
38151 38527.66667 38897.33333 39226.66667 39526 39766.66667
40058.33333 40349.33333 40554.66667 40759.66667 40977.66667 41150.66667
41325.66667 41518.66667 41606.66667 41777.33333 41845.33333 41977.66667
42015.33333 42116.33333 42137.66667 42136.33333 42185.66667 42155.66667
42124.33333 42087 42054.66667 42014 41959.66667 41903.33333
41832.33333 41778.33333 41708.66667 41663.66667 41638.33333 41598.66667
41550 41492 41424.66667 41374 41335 41285 41201.66667
41086.33333 40944.33333 40771.33333 40616.33333 40496.66667 40428.66667
40396 40394.33333 40426.66667 40416.33333 40415 40402.33333 40282
40197.33333 40205 40252 40439.66667 40517.66667 40314.33333
40145.33333 40132 40195 40234.33333 39989 39892 40112.33333
40045.33333 39964.33333 39701.66667 39329.33333 39101.33333 38771.66667
38443.33333 37884.66667 37163.33333 36779.33333 36912 36586 36052
35758.66667 34737 33775.66667 33285 32825 31775.33333
30978.33333];
elseif w==3
    r=.3;
    %.3(3) - can truncate at t= 90.275, exp= 31274
    timedata=[0 0.01 0.02 0.03 0.04 0.05 0.06
0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15
0.16 0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.24
0.25 0.26 0.27 0.28 0.3 0.335 0.375 0.42 0.475
0.535 0.605 0.685 0.77 0.87 0.985 1.115 1.26 1.42
1.605 1.815 2.05 2.315 2.615 2.955 3.335 3.765 4.255
4.81 5.435 6.14 6.935 7.835 8.855 10.005 11.305 12.775
14.435 16.31 18.43 20.825 23.535 26.595 30.05 33.955 38.37
43.36 48.995 55.365 62.565 70.7 79.89 90.275 102.01 115.27
130.25 147.19 166.33 199.43 225.35 254.65 287.76 325.16 367.44
415.21 469.2 530.21 599.14 677.03 765.05 864.51 976.91 1103.9
1247.5 1409.6 1592.9 1800];
    expdata=[0 52.448 478.99 1133.5 1791.7 2420.8 3005.5
3543 4016.5 4470.2 4877.3 5276.1 5615.4 5991.2 6341.7 6650.9
6953.1 7268.9 7544.4 7821.5 8081.8 8344.2 8591.3 8818.4 9049.9
9282.1 9508.5 9704.7 9931 10338 10991 11662 12372 13169
13978 14883 15810 16667 17568 18476 19497 20560 21509
22470 23427 24406 25296 26267 27183 28017 28770 29596
30311 30940 31554 32044 32436 32770 33007 33208 33339
33389 33404 33326 33194 33039 32857 32667 32459 32261
32213 32138 31999 31842 31653 31466 31274 31157 31099
30875 30788 30765 30926 30888 30724 30581 30448 30265
30404 30118 30184 29777 29055 29118 28672 26800 27125
26434 25741 24756 24923];
elseif w==4
    r=1;
    %1(2) - can truncate at t= 90.275, exp= 15891
    timedata=[0 0.01 0.02 0.03 0.04 0.05 0.06
0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15
0.16 0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.24
0.25 0.26 0.27 0.28 0.3 0.335 0.375 0.42 0.475
0.535 0.605 0.685 0.77 0.87 0.985 1.115 1.26 1.42
1.605 1.815 2.05 2.315 2.615 2.955 3.335 3.765 4.255
4.81 5.435 6.14 6.935 7.835 8.855 10.005 11.305 12.775
14.435 16.31 18.43 20.825 23.535 26.595 30.05 33.955 38.37
43.36 48.995 55.365 62.565 70.7 79.89 90.275 102.01 115.27

```

```

130.25 147.19 166.33 199.43 225.35 254.65 287.76 325.16 367.44
415.21 469.2 530.21 599.14 677.03 765.05 864.51 976.91 1103.9
1247.5 1409.6 1592.9 1800];
    expdata=[0 65.549 490.9 1090.5 1764.7 2393.7 2976.9
3507.7 4000.8 4447.5 4881.1 5267.9 5634.4 5980.8 6320.5 6629
6936.4 7220 7492.1 7762.9 8023.9 8264.7 8537.8 8768.7 9012.4
9243.8 9522 9742.7 9965 10344 11064 11769 12531 13337
14182 15000 15918 16792 17643 18543 19392 20174 20862
21528 22074 22543 22862 23075 23141 23055 22863 22522
22081 21608 21144 20731 20367 20097 19882 19660 19446
19275 19245 19164 19095 18929 18594 18297 18305 18445
18175 17508 17196 16573 16211 16175 15891 16108 15234
15704 15690 15466 15594 15998 15671 15906 16899 17104
17162 17846 17468 16644 16772 16497 16022 16569 16323
16064 17184 16867 16103];
elseif w==5
    r=3;
    %ave data - can truncate at t= 90.275, exp= 10376
    timedata=[0.01 0.02 0.03 0.04 0.05 0.06
0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15
0.16 0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.24
0.25 0.26 0.27 0.28 0.3 0.335 0.375 0.42 0.475
0.535 0.605 0.685 0.77 0.87 0.985 1.115 1.26 1.42
1.605 1.815 2.05 2.315 2.615 2.955 3.335 3.765 4.255
4.81 5.435 6.14 6.935 7.835 8.855 10.005 11.305 12.775
14.435 16.31 18.43 20.825 23.535 26.595 30.05 33.955 38.37
43.36 48.995 55.365 62.565 70.7 79.89 90.275 102.01 115.27
130.25 147.19 166.33 199.43 225.35 254.65 287.76 325.16 367.44
415.21 469.2 530.21 599.14 677.03 765.05 864.51 976.91 1103.9
1247.5 1409.6 1592.9 1800];
    expdata=[80.791 534.02 1183.7 1868.1 2498.3 3107.3
3649.5 4140 4589.9 5004.1 5402.2 5790.6 6128 6480 6793.6
7078.1 7388.7 7651.8 7926.1 8174.3 8411 8666 8901.7 9077.6
9313 9485.5 9716.8 9897.5 10246 10841 11437 12009 12622
13104 13618 14041 14366 14623 14761 14772 14695 14522
14212 13892 13525 13147 12768 12443 12152 11859 11617
11382 11214 11239 11207 11171 11208 11217 11184 11060
10992 10824 10663 10649 10602 10652 10623 10612 10545
10506 10449 10222 10274 10306 10286 10376 10438 10567
10267 10072 10103 10039 9804.6 9809.6 9994.7 9855.8 9974.2
9955.5 9543.2 9700.2 9800.7 9600.7 9304 9541 9758.7 9939.9
10043 8959.4 9158.9 9522.6];
end

%shift time data back to account for start up delay in
rheometer
factor=timedata(2)-0.0005;
timedata=timedata-factor;
timedata(1)=0;

%differential equation solver to get model prediction of
viscosity vs time

[t,etatemp,diverge]=shear_pde_solver(r,wt,sigma,CI,para(1,:),para(2,:),
para(3,:),timedata,modes,re,orient);
if diverge==1 %keep track of divergence

```

```

        fprintf('Divergent Point (shear):')
        fprintf('\nLambda: %f',para(2,:))
        fprintf('\nAlpha: %f\n\n',para(3,:))
    elseif diverge==0 %if no divergence, calculate error between
model and experiment
        for k=2:length(timedata)
            Serror(w)=Serror(w) + (log10(expdata(k))-
log10(etatemp(k)))^2;
        end
    end
end

%plot the model predictions and experimental data
if graph==1
    if w==1
        figure
        loglog(timedata,expdata,'*',t,etatemp,'Color',[1,0,0]);
        hold on
    elseif w==2
        loglog(timedata,expdata,'*',t,etatemp,'Color',[0,1,0]);
    elseif w==3
        loglog(timedata,expdata,'*',t,etatemp,'Color',[0,0,1]);
    elseif w==4
        loglog(timedata,expdata,'*',t,etatemp,'Color',[0,0,0]);
    elseif w==5
        loglog(timedata,expdata,'*',t,etatemp,'Color',[0,1,1]);
        xlabel('Time (s)')
        ylabel('Viscosity (Pa*s)')
        title('Shear Viscosity')
        legend('Exp \gamma=0.01','Mod \gamma=0.01','Exp
\gamma=0.1','Mod \gamma=0.1','Exp \gamma=0.3','Mod \gamma=0.3','Exp
\gamma=1','Mod \gamma=1','Exp \gamma=3','Mod \gamma=3',-1);
        set(gcf,'Units','normalized',
'WindowStyle','docked','OuterPosition',[0 0 1 1]); %this places the
plot in a maximized window
        hold off
    end

end
end
Serror=sum(Serror); %sum up error from all shear rates
end

```


Pure Polymer: Transient Extensional Flow

```
function [Terror] = Extensional_MB_0wt(para,graph,modes)
%This program calls on extensional_pde_solver to solve the transient
%differential equations from the constitutive model and create a model
%prediction for the transient extensional viscosity vs time. The error
%between this model prediction and the experimental data is calculated
and
%sent back to Pure_Polymer_Optimization to be used in the error
minimization
%for parameter optimization. This program can also plot the model
%prediction of transient extensional viscosity given the set of pure
%polymer parameters

wt=0; %mass fraction of cnfs
sigma=1; %no cnfs - value doesn't matter
CI=0; %no cnfs - value doesn't matter
re=0; %no cnfs - value doesn't matter
orient=[0 0 0]; %no cnfs - value doesn't matter

Terror=zeros(1,5); %initialize error variable

for m=1:5 % m designates the m-th extension rate
    if m==1
        r=.01; %extensional rate
        %data Koki used at 900s melt time 0.01(4) - TRUNCATED
        timedata= [0 38.2077 45.7507 54.7829 65.5982 78.5487
112.6245 134.8589 161.4830 193.3631 231.5372 277.2475
331.9821 397.5225 476.0019 569.9748 682.5000];
        expdata=[0 168492.5 170607.5 175964.1 179823.5
183979.9 199236.2 204547.3 226705.7 262619 314207.9
403477 602314.5 1129767 2432705 6232116 20593550];
    elseif m==2
        r=.03;
        %data Koki used at 400s melt time 0.03(4) - TRUNCATED
        timedata=[0 11.35781 13.29856 15.57092 18.23158
21.34686 24.99447 29.26535 34.26601 40.12115 46.97677
55.00384 64.40251 75.40717 88.29222 103.379 141.7268
165.9441 194.2995];
        expdata=[0 146779.5000 148095.4000 154832.9000 159295.1000
161106.7000 163667.9000 169304.2000 176864.4000 180235.5000 186648.7000
198452.5000 213460.4000 231445.1000 259906.8000 294510.1000 499871.8000
1082033.0000 2616750.0000];
    elseif m==3
        r=.1;
        %data set Koki used at 400s melt time 0.1(3)
        timedata = [0 3.190139 3.644588 4.163775 4.756922
5.434566 6.208744 7.093206 8.103663 9.258065 10.57692
12.08364 13.80501 15.77159 18.01832 20.58511 23.51755
26.86772 30.69514 45.77055 52.29077 59.73981 68.25];
        expdata = [0 113170.4000 115866.3000 115710.9000
119300.8000 128344.5000 132105.2000 139148.5000 146302.6000 151249.3000
157990.5000 165545.4000 174792.1000 182221.3000 193300.1000 200885.4000
210597.0000 222691.6000 250768.3000 380212.9000 444930.5000 626399.9000
1159610.0000];
    end
end
```

```

elseif m==4
    r=.3;
    %data set Koki used at 900s melt time 0.3(4)
    timedata = [0 0.1 0.1117125 0.1247968 0.1394135
0.1557423 0.1739836 0.1943614 0.2171259 0.2425567 0.2709661
0.3027029 0.3381569 0.3777634 0.4220089 0.4714366 0.5266534
0.5883376 0.6572464 0.7342262 0.8202223 0.9162906 1.023611
1.143501 1.277433 1.427052 1.594195 1.780915 1.989504
2.222524 2.482837 2.773638 3.0985 3.461411 3.866828
4.319729 4.825676 5.390882 6.022287 6.727646 7.51562
8.395884 9.37925 10.47779 11.705];
    expdata=[0 9063.837 9737.7 10258.08 12387.07
18360.81 23822.21 25218.44 29457.06 33695.11 33500.95
38132.72 43247.05 44498.33 49870.7 53450.51 58368.96
63280.75 66423.27 70025.27 73785.37 68276.05 75092.21
70893.91 74983.12 79406.44 84650.98 89659.12 94508.59
98780.58 103107.8 108275.5 114432 121238.7 127093.3
132984.4 140180.4 144799.7 156472.5 163388.1 180290.9
192133.9 215290.3 255532.4 310504];
elseif m==5
    r=1;
    %data set Koki used at 400s 1.0(2)
    timedata = [0 0.1000 0.1090 0.1188 0.1295 0.1412
0.1539 0.1677 0.1828 0.1993 0.2172 0.2368 0.2581 0.2813 0.3066
0.3342 0.3643 0.3971 0.4328 0.4718 0.5143 0.5605 0.6110 0.6660
0.7259 0.7913 0.8625 0.9402 1.0248 1.1170 1.2176 1.3272 1.4466
1.5768 1.7187 1.8735 2.0421 2.2259 2.4262 2.6446 2.8827 3.1421
3.4250 3.7333 4.0693 4.4356 4.8348 5.2700 5.7444 6.2614 ];
    expdata = [0 8603.345 10275.25 11740.97 13521.02
16231.83 20042.37 24496.71 25221.1 26805.39 29678.07
33029.05 35903.96 39329.07 40819.31 38128.38 42085.04
44251.41 46737.63 49700.84 53122.45 55426.59 59770.41
63570.79 67420.58 71928.44 76829.28 82574.72 88348.64
94573.99 100835 107093.4 114905.7 123918.8 133751.8
145084.7 159288 173989.3 192173.7 212003.2 245028.6
288546.2 334680.5 379258.3 425686.6 486103.8 568304.4
626113.8 673079.6 803106.5];
end

% differential equation solver to get model prediction of viscosity
vs time

[t,etatep,diverge]=extensional_pde_solver(r,wt,sigma,CI,para(1,:),para
(2,:),para(3,:),timedata,modes,re,orient); %use for calculation of
Error
if diverge==1 %keep track of divergence
    fprintf('Divergent Point (ext):')
    fprintf('\nLambda: %f',para(2,:))
    fprintf('\nAlpha: %f\n\n',para(3,:))
elseif diverge==0 %if no divergence, calculate error between model
and experiment
    for k=2:length(timedata)
        Error(m)=Error(m) + (log10(expdata(k))-
log10(etatep(k)))^2;
    end
end
end

```

```

%plot the model predictions and experimental data
if graph==1
    if m==1
        loglog(t,etatemp,'r',timedata,expdata,'*r');
        hold on
    elseif m==2
        loglog(t,etatemp,'g',timedata,expdata,'*g');
    elseif m==3
        loglog(t,etatemp,'b',timedata,expdata,'*b');
    elseif m==4
        loglog(t,etatemp,'c',timedata,expdata,'*c');
    elseif m==5
        loglog(t,etatemp,'k',timedata,expdata,'*k');
        title('Extensional Viscosity');
        xlabel('Time (s)');
        ylabel('Viscosity (Pa*s)');
        legend('Mod \epsilon=.01','Exp \epsilon=.01','Mod
\epsilon=.03','Exp \epsilon=.03','Mod \epsilon=.1','Exp
\epsilon=.1','Mod \epsilon=.3','Exp \epsilon=.3','Mod \epsilon=1','Exp
\epsilon=1',-1);
        set(gcf,'Units','normalized',
'WindowStyle','docked','OuterPosition',[0 0 1 1]); %this places the
plot in a maximized window
        hold off
    end
end

end

Error=sum(Error); %sum up error from all extension rates
end

```

Determining Boundary of Convergence for σ , λ , and α

```
function Parameter_Convergence
%This program tests convergence of model due to alphas and lambdas for a
%range of sigmas
%By Tim's theory:
%  tests at the most extreme: highest deformational rates, highest wt%
%  single mode
%give the boundary of the parameter convergence

warning('off','all')

%optimized pure polymer parameters
Final=[ 0.961768502242416e+4 0.849997471090374e+4
0.141393472087865e+4 1.782368034425766e+4 2.400790731282941e+4;
%etap by mode
      9.789278339157816e+3 0.000189405013332e+3
0.000013348257373e+3 0.018781148809731e+3 0.001901628993121e+3;
%lambda by mode
      0.001603491662313 0.985105739933884 0.847316738754344
0.132461526978848 0.554616118726100]; %alpha by mode

%values pulled out of 'Final' matrix
eta=[0.961768502242416e+4 0.849997471090374e+4 0.141393472087865e+4
1.782368034425766e+4 2.400790731282941e+4];
lambda=[9.789278339157816e+3 0.000189405013332e+3
0.000013348257373e+3 0.018781148809731e+3 0.001901628993121e+3];
alpha=[0.001603491662313 0.985105739933884 0.847316738754344
0.132461526978848 0.554616118726100];

sigma=.1:.1:1; %range of sigmas to test at
CI=.09;
wt=.1; %highest weight percent of interest
shear_rate=3; %highest shear rate of interest
shear_time=[0 0.01 0.02 0.03 0.04 0.05 0.06 0.07
0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15 0.16
0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.25 0.285
0.325 0.37 0.425 0.49 0.565 0.65 0.75 0.865 0.995
1.145 1.315 1.51 1.74 2.005 2.31 2.66 3.06 3.525
4.06 4.675 5.38 6.195 7.135 8.215 9.46 10.895 12.545
14.445 16.635 19.155 22.06 25.41 29.265 33.7 38.81 44.695
51.475 59.285 68.275 78.63 90.555 104.29 120.11 138.32 170.54
196.39 226.18 260.49 300.01];
shear_eta=[0 128.86 1006 1343.5 3486.7 4705.7 5808.4 6804.9
7699.4 8541.3 9324.6 9990.2 10618 11210 11803 12262 12708
13230 13703 14069 14486 14777 15181 15508 16140 17072
17926 18725 19471 20077 20517 20791 20771 20384 19659
18649 17570 16732 16039 15622 15359 15062 14832 14575
14577 14929 15216 15221 15140 15024 15021 15042 14762
13992 13644 13652 13425 13484 12779 11962 11177 10332
11237 11004 9504.5 9399.4 8319 7674.5 6953 6238.7 5759.6
5584.8 5409.9 5322.7 5310.3];
ext_rate=1; %highest extension rate of interest
```

```

ext_time=[0 0.1 0.109001 0.1188123 0.1295066 0.1411636
0.1538698 0.1677197 0.1828162 0.1992716 0.2172081 0.2367591
0.2580699 0.2812989 0.3066187 0.3342176 0.3643007 0.3970916
0.432834 0.4717935 0.5142599 0.5605487 0.6110039 0.6660007
0.7259477 0.7912906 0.862515 0.9401504 1.024774 1.117014
1.217557 1.32715 1.446607 1.576817];
ext_eta=[0 20591.25 27356.84 32020.18 39944.88 49080.84
56677.08 61538.6 65226.97 70765.52 77795.37 82053.54
86141.62 93525.47 96412.62 96613.42 104997.7 108339.2
116050.4 120507.6 129031.3 134957.4 142610 150228
157416.4 164235.9 170554.6 177867.3 183905.5 187905
191496.6 196336.7 199949.9 200844.6];
p=1; q=1; conv1=zeros(100,3); conv2=zeros(100,3);

%for-loop to test the boundary at all sigma values
for i=1:10
    for j=1:8
        for k=1:100
            lambda(5)=10^(j-3);
            alpha(5)=.001*k;
            [t1
etatempl,flag1]=ext_pde(eta,lambda,alpha,sigma(i),CI,wt,ext_rate,ext_time);

            flag1
            if flag1(2)>10
                conv1(p,:)=[i j k];
                p=p+1;
            end
            eta=[0.961768502242416e+4 0.849997471090374e+4
0.141393472087865e+4 1.782368034425766e+4 2.400790731282941e+4];
            [t2
etatempl,flag2]=shr_pde(eta,lambda,alpha,sigma(i),CI,wt,shear_rate,shear_time);

            flag2
            if flag2(2)>10
                conv2(q,:)=[i j k];
                q=q+1;
            end
        end
    end
end
conv1
conv2
plot(conv1(2),conv1(3))
figure
plot(conv2(2),conv2(3))

%write results to excel spreadsheet
xlswrite('ParamConverg',conv1,'Sheet1');
xlswrite('ParamConverg',conv2,'Sheet2');

function [time, etac,flag] =
ext_pde(eta,lambda,alpha,sigma,CI,mass,r,tspan)
%this subfunction is a specialized version of 'extensional_pde_solver'
that

```

```
%examines divergence. see that program for specific details about its
%operation
```

```
x=1;
total12=0;
y=zeros(1,3);
tp=zeros(1,3);
re=44.0;
chi=1.0*(re^2-1)/(re^2+1); %another form of aspect ratio
rf=1750.0; %fiber density
rs=1000.0; %polymer density
```

```
phi=rs*mass/(rf+(rs-rf)*mass); %volume fract of CNFs
Ap=re^2/(3*log(sqrt(pi/phi))); %shape factor, aka A_2
```

```
orientation=[.586, .207, .207];
```

```
initial=[0 0 0 orientation];
while x<6
    options=odeset('Stats','on');
    [time, yout,
flag]=ode23tb(@modeextensionalsub,tspan,initial,options);
```

```
    tau11=yout(:,1);
    tau22=yout(:,2);
    tau12=(tau11-tau22);
    asdf=size(tau12)
    if asdf==34
        total12=total12+tau12;
    elseif asdf~=34
        break
    end
```

```
    a11=yout(:,4);
    a22=yout(:,5);
    a33=yout(:,6);
```

```
    x=x+1;
end
```

```
eta=total12./r;
coef=2.0.*eta*phi;
tf12=(coef).*(Ap*r*((27.0*(a11.*a22.*a33)).*(-3/35+(4*a11+2*a22)/7)+(1-
27*(a11.*a22.*a33)).*(a11.^2+(-3.*a11.*a22-
a11.*a33+a22.^2+a22.*a33)/2)));
etac=tf12/r+eta;
```

```
function dy = modeextensionalsub(t,y)
```

```
    t11=y(1);
    t22=y(2);
    t33=y(3);
    a11=y(4);
    a22=y(5);
```

```

a33=y(6);

dy=zeros(6,1);
dy(1,1)=2*r*etap(x)/lambda(x)-sigma*t11/lambda(x)-
alpha(x)/etap(x)*t11^2+2*r*t11-3*(1-sigma)*2*a11*t11/lambda(x);
dy(2,1)=-r*etap(x)/lambda(x)-sigma*t22/lambda(x)-
alpha(x)/etap(x)*t22^2-r*t22-3*(1-sigma)*a22*t22/lambda(x);
dy(3,1)=-r*etap(x)/lambda(x)-sigma*t33/lambda(x)-
alpha(x)/etap(x)*t33^2-r*t33-3*(1-sigma)*a33*t33/lambda(x);

dy(4,1)=CI*2*3^(.5)*r*(1-3*a11)+2*chi*r*a11-
2*chi*r*(27*a11*a22*a33*(-2/35+(10*a11-a22-a33)/14)+(1-
27*a11*a22*a33)*(a11^2-(a11*a22+y(1)*a33)/2));
dy(5,1)=CI*2*3^(.5)*r*(1-3*a22)-chi*r*a22-
2*chi*r*(27*a11*a22*a33*(1/35+(2*a11-5*a22-a33)/14)+(1-
27*a11*a22*a33)*(a11*a22-(a22^2+a22*a33)/2));
dy(6,1)=CI*2*3^(.5)*r*(1-3*a33)-chi*r*a33-
2*chi*r*(27*a11*a22*a33*(1/35+(2*a11-a22-5*a33)/14)+(1-
27*a11*a22*a33)*(a11*a33-(a22*a33+a33^2)/2));

end

end

function [time1, etaf,flag] =
shr_pde(etap,lambda,alpha,sigma,CI,mass,r,tspan)
%this subfunction is a specialized version of 'shear_pde_solver' that
%examines divergence. see that program for specific details about its
%operation

modes=5;
total12=0;
y=zeros(1,3);
tp=zeros(1,3);
re=44.0; %aspect ratio, aka h %keep this value in mind

chi=1.0*(re^2-1)/(re^2+1); %another form of aspect ratio
rf=1750.0; %fiber density
rs=1000.0; %polymer density

phi=rs*mass/(rf+(rs-rf)*mass); %volume fract of CNFs
Ap=re^2/(3*log(sqrt(pi/phi))); %shape factor, aka A_2

orientation=[.586, .207, .207];
initial=[0 0 0 0 orientation];

tau_finalf=zeros(4,5);

for x=1:modes
initial=[0 0 0 0 orientation]; %initial conditions for ode solver

options=odeset('Stats','on');

```

```

    [time1, yo, flag]=ode23tb(@modeshearsub,tspan,initial,options);
%ode solver for solving Equations (2) & (4) simultaneously

    L=length(time1);

    tau1=zeros(L,5);
    tau12=zeros(L,5);
    tau2=zeros(L,5);
    tau3=zeros(L,5);

    tau1(:,x)=yo(:,1);
    tau12(:,x)=yo(:,2);
    tau2(:,x)=yo(:,3);
    tau3(:,x)=yo(:,4);

    tau_finalf(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x) tau3(L,x)]';

end

total11f=sum(tau1,2);
total12f=sum(tau12,2);
total22f=sum(tau2,2);
total33f=sum(tau3,2);

a11f=yo(:,5);
a12f=yo(:,6);
a22f=yo(:,7);
a33f=1-a11f-a22f;

T_final= [total11f(L) total12f(L) total22f(L) total33f(L)];

a_final = [a11f(L) a12f(L) a22f(L)]';

etaf=total12f./r;
coef=2.0*etaf*phi;

%%%This section computes the fiber stress from Equation (3)
if r~=0
    %disp('r~=0')
    tf12f=(coef*Ap*r).*((27.0.*(a11f.*a22f.*(1-a11f-a22f)-
    (a12f.^2).*(1-a11f-a22f))).*(-
    1.0/35.0+1.0/7.0.*(a11f+a22f)))+(a12f.^2).*(1-27.0.*(a11f.*a22f.*(1-a11f-
    a22f))+27.0.*(a12f.^2).*(1-a11f-a22f)));
    tf11f=(coef*Ap*r).*((27.0.*(a11f.*a22f.*(1-a11f-a22f)-
    (a12f.^2).*(1-a11f-a22f))).*(3.0/7.0.*(a12f)))+(a11f.*a12f).*(1-
    27.*(a11f.*a22f.*(1-a11f-a22f))+27.0.*(a12f.^2).*(1-a11f-a22f)));
    tf22f=(coef*Ap*r).*((27.0.*(a11f.*a22f.*(1-a11f-a22f)-
    (a12f.^2).*(1-a11f-a22f))).*(3.0/7.0.*(a12f)))+(a22f.*a12f).*(1-
    27.*(a11f.*a22f.*(1-a11f-a22f))+27.0.*(a12f.^2).*(1-a11f-a22f)));
    tf33f=(coef*Ap*r).*((27.0.*(a11f.*a22f.*(1-a11f-a22f)-
    (a12f.^2).*(1-a11f-a22f))).*(1.0/7.0.*(a12f)))+(1-a11f-

```



```

a22f).*a12f)).*(1-27.*(a11f.*a22f.*(1-a11f-a22f))+27.0.*(a12f.^2)).*(1-
a11f-a22f)));
else
    %disp('r==0')
    tf12f=(2.0*phi.*total12f.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-
(a12f.^2)).*(1-a11f-a22f))).*(-
1.0/35.0+1.0/7.0*(a11f+a22f)))+(a12f.^2)).*(1-27.*(a11f.*a22f.*(1-a11f-
a22f))+27.0*(a12f.^2)).*(1-a11f-a22f)));
    tf11f=(2.0*phi.*total12f.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-
(a12f.^2)).*(1-a11f-a22f))).*(3.0/7.0*(a12f)))+(a11f.*a12f)).*(1-
27*(a11f.*a22f.*(1-a11f-a22f))+27.0*(a12f.^2)).*(1-a11f-a22f)));
    tf22f=(2.0*phi.*total12f.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-
(a12f.^2)).*(1-a11f-a22f))).*(3.0/7.0*(a12f)))+(a22f.*a12f)).*(1-
27*(a11f.*a22f.*(1-a11f-a22f))+27.0*(a12f.^2)).*(1-a11f-a22f)));
    tf33f=(2.0*phi.*total12f.*Ap).*((27.0*(a11f.*a22f.*(1-a11f-a22f)-
(a12f.^2)).*(1-a11f-a22f))).*(1.0/7.0*(a12f)))+(1-a11f-a22f).*a12f)).*(1-
27*(a11f.*a22f.*(1-a11f-a22f))+27.0*(a12f.^2)).*(1-a11f-a22f)));
end

tauc12f=tf12f+total12f;
tauc11f=tf11f+total11f;
tauc22f=tf22f+total22f;
tauc33f=tf33f+total33f;
etacf=tauc12f./r;

function dyo = modeshearsub(t,y)

    tp11=y(1);
    tp12=y(2);
    tp22=y(3);
    tp33=y(4);
    a11=y(5);
    a12=y(6);
    a22=y(7);
    a33=1-a11-a22;
    dyo=zeros(7,1);

    dyo(1,1)=-alpha(x)*(tp11^2+tp12^2)/etap(x)-
sigma*tp11/lambda(x)+2*r*tp12-3*(1-
sigma)*(tp11*a11+tp12*a12)/lambda(x);
    dyo(2,1)=-alpha(x)*(tp11*tp12+tp12*tp22)/etap(x)-
sigma*tp12/lambda(x)+etap(x)*r/lambda(x)+r*tp22-3*(1-
sigma)/2/lambda(x)*(a11*tp12+a12*tp22+a12*tp11+a22*tp12);
    dyo(3,1)=-alpha(x)*(tp12^2+tp22^2)/etap(x)-
sigma*tp22/lambda(x)-3*(1-sigma)/lambda(x)*(tp12*a12+tp22*a22);
    dyo(4,1)=-alpha(x)*(tp33^2)/etap(x)-tp33*sigma/lambda(x)-3*(1-
sigma)/lambda(x)*a33*tp33;

    dyo(5,1)=r*a12+2*CI*abs(r)*(1.0-3.0*a11)+chi*r*a12-...
        2*chi*r*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)...
        -27.0*(a12^2)*(1-a11-a22)))+(1.0-27.0*a11*a22*(1-a11-
a22))+...
        27.0*(a12^2)*(1-a11-a22))*a11*a12);
    dyo(6,1)=1.0/2.0*r*a22-1.0/2.0*r*a11+chi*((1.0/2.0*r*a22+...
        1.0/2.0*r*a11)-(2.0*r)*((27.0*a11*a22*(1-a11-a22)-...

```

```

                27.0*(a12^2)*(1-a11-a22))*(-
1.0/35.0+1.0/7.0*a11+1.0/7.0*a22)+...
                (1.0-27.0*a11*a22*(1-a11-a22)...
+27.0*(a12^2)*(1-a11-a22))*a12^2))-6.0*CI*abs(r)*a12;
dyo(7,1)=-r*a12+2*CI*abs(r)*(1.0-3.0*a22)+chi*r*a12-...
                2*chi*r*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)-...
27.0*(a12^2)*(1-a11-a22)))+(1.0-27.0*a11*a22*(1-a11-a22)+...
27.0*(a12^2)*(1-a11-a22))*a12*a22);

end

end

end

```

Composite Parameter Optimization: C_i

```
function [ci_mb2,ci_mb2hht,ci_sc2]=ci_optimization()
%This program is used for optimizing the values of CI for MB2, MB2HHT,
and
%SC2 composites using the steady state orientation of cnfs (all
component)
%in extensional flows. Orientation at one or two extension rates are
used
```

```
f0=[.01]; %initial guess for fmincon
options=optimset('Algorithm','interior-point'); %search algorithm
```

```
%MB2
[ci_mb2, fval,
exitflag]=fmincon(@MB2,f0,[],[],[],[],[0],[1],[],options);
```

```
function e=MB2(ci)
%this sub-function calculates error between model and experiment
for
%fmincon so fmincon can optimize ci
re=53;
strain_rate=[.01,.1,1];
all_mb2_exp=[.928 .94 .94]; %experimental orientation (all)
all_mb2=zeros(1,3);
for i=1:length(strain_rate)
t0=[0,3/strain_rate(i)];
%calculate model prediction
[t,a]=ode45(@orient,t0,[.586 .207 .207 re ci
strain_rate(i)]);
all_mb2(i)=a(end,1); %model prediction for steady-state
value
end
e=sum((all_mb2_exp-all_mb2).^2); %total error
end
```

```
%MB2-HHT
[ci_mb2hht, fval,
exitflag]=fmincon(@MB2HHT,f0,[],[],[],[],[0],[1],[],options);
```

```
function e=MB2HHT(ci)
%this sub-function calculates error between model and experiment
for
%fmincon so fmincon can optimize ci
re=44;
strain_rate=[.1];
all_mb2hht_exp=[.937]; %experimental orientation (all)
all_mb2hht=zeros(1,1);
for i=1:length(strain_rate)
t0=[0,3/strain_rate(i)];
%calculate model prediction
[t,a]=ode45(@orient,t0,[.441 .2795 .2795 re ci
strain_rate(i)]);
all_mb2hht(i)=a(end,1); %model prediction for steady-state
value
```

```

        end
        e=sum((a11_mb2hht_exp-a11_mb2hht).^2); %total error
    end

%SC2
[ci_sc2, fval,
exitflag]=fmincon(@SC2,f0,[],[],[],[],[0],[1],[],options);

function e=SC2(ci)
%this sub-function calculates error between model and experiment
for
    %fmincon so fmincon can optimize ci
    re=69;
    strain_rate=[.01,.1];
    a11_sc2_exp=[.909 .89]; %experimental orientation (a11)
    a11_sc2=zeros(1,2);
    for i=1:length(strain_rate)
        t0=[0,3/strain_rate(i)];
        %calculate model prediction
        [t,a]=ode45(@orient,t0,[.528 .236 .236 re ci
strain_rate(i)]);
        a11_sc2(i)=a(end,1); %model prediction for steady-state
value
    end
    e=sum((a11_sc2_exp-a11_sc2).^2); %total error
end

%display optimized values
ci_mb2;
ci_mb2hht;
ci_sc2;

function dy = orient(t,y)
%this subfunction solves the differential equation that describes
the
%orientation evolution of cnfs
    a11e=y(1);
    a22e=y(2);
    a33e=y(3);
    chi=1.0*(y(4)^2-1)/(y(4)^2+1);
    CI=y(5);
    r=y(6);
    dy=zeros(6,1);

    %extensional orientation evolution equations
    dy(1,1)=CI*2*3^(.5)*r*(1-3*a11e)+2*chi*r*a11e-
2*chi*r*(27*a11e*a22e*a33e*(-2/35+(10*a11e-a22e-a33e)/14)+(1-
27*a11e*a22e*a33e)*(a11e^2-(a11e*a22e+a11e*a33e)/2));
    dy(2,1)=CI*2*3^(.5)*r*(1-3*a22e)-chi*r*a22e-
2*chi*r*(27*a11e*a22e*a33e*(1/35+(2*a11e-5*a22e-a33e)/14)+(1-
27*a11e*a22e*a33e)*(a11e*a22e-(a22e^2+a22e*a33e)/2));
    dy(3,1)=CI*2*3^(.5)*r*(1-3*a33e)-chi*r*a33e-
2*chi*r*(27*a11e*a22e*a33e*(1/35+(2*a11e-a22e-5*a33e)/14)+(1-
27*a11e*a22e*a33e)*(a11e*a33e-(a22e*a33e+a33e^2)/2));
    end
end

```

Composite Parameter Optimization: Aspect Ratio Scaling Factor

```
function
[ scale_mb2, scale_mb2hht, scale_sc2 ] = aspect_ratio_optimization(z, CI)
%This function optimizes the scaling factor for aspect ratio for MB2,
%MB2HHT, and SC2 for given values of sigma. These results can then be
used
%in "sigma_optimization" to optimize sigma for these scaling factors.
%Iteratively use these two programs until the values for the scaling
%factors and sigmas converge

sig = [.8099 .5439 .5]; %change with iterations with "sigma_optimization"

CI = [.0306, .0301, .0499]; %optimized values from "ci_optimization"
program
z = .01; %relative error scaling (make error from shear and from
extensional same order of magnitude)

%optimized parameters from pure polymer
etap = [1563.180023 10503.23358 30793.85272 9864.433234 9310.030642
40690.01669];
lambda = [0.01461114 0.235979496 3.210427777 34.39722848 6390.005373
116255.5053];
alpha = [0.818868357 0.56 0.56 0.078601046 0.025189799
0.001841609];
rate = [.1, .3, 1, 3]; %shear rates
wt = .02; %mass fraction of cnfs
tspan = 0:100; %time span for differential equation solver
modes = 6;

f0 = 1; %initial guess for fmincon
options = optimset('Algorithm', 'interior-point'); %search algorithm

%MB2
for i = 1:4
    %experimental viscosity
    if i == 1
        expdata = [0 50.7945 584.48 1344.55 2156.55 2936.75 3642.3
4300.35 4916.1 5486.55 6033.9 6547.35 6989.2 7478.75 7894.65 8337.05
8711.15 9078.3 9467.1 9817.15 10128.55 10489.5 10776 11133.5
11398 11718.5 11999 12539.5 13462 14433 15452 16596 17857
19195.5 20482 21804.5 23187.5 24604 26029 27489 28942.5 30349
31871 33540 35291 36825.5 38329 39883.5 41336.5 42792 44163.5
45455.5 46928 48050 49140.5 50051.5 50998.5 51635.5 52298 52613.5
52941.5 52956 52840 52555 52109 51529 50807.5 50015 49252.5
48562.5 47945.5 47447.5 47096 46859.5 46687.5 46553 46500.5 46521.5
46502 46310 46213 46145 46097.5 46524 46700 46803 46926
46665 46758.5 46678.5 46505.5 45537.5 44098 43581 41589.5 40157.5
38101.5 36936.5 34339 32735 30810];
    elseif i == 2
        expdata = [0 99.9685 604.9975 1340.875 2132.75 2901.175
3615.725 4250.8 4849.225 5396.025 5902.575 6368.925
6840.575 7264.625 7669.825 8073.225 8450.65 8810.425
9167.825 9500.225 9835.875 10147.425 10466.975 10761.575
11107.75 11687.75 12541.75 13574.25 14676 15808 17049
```

```

18279.75    19525.75    20877.5 22255    23687    25110.75    26524
27946.75    29342.75    30698    32143.5 33463.25    34690.75    35815.5
36905.75    37777    38536.75    39114.5 39512    39696.75    39695
39474.25    39051.75    38471    37797    37087.5 36418.25    35882
35513.25    35195.25    34831.25    34404.5 34004.5 33589.5 33141.75
32648.25    32318.75    32091.5 31819    31350.5 30692    30481.25
30675.25    30402.25    29974.5 29579.5 28524.75    27139.75    26114.5
24383.5 23727.25    22031.25    21686.5 20093.75    19055];
    elseif i==3
        expdata=[0 75.508    584.995 1305.55 2089.8    2826.1    3530.25
4155.5    4741.3    5265.55 5766.4    6218.95 6653.3    7066.3    7452.2    7830.9
8208.15    8569.3    8903.4    9235.3    9574    9888.8    10211.55    10576.5
11194    12001.5 12934.5 13984    15073    16183    17332    18511    19647
20766.5 21847.5 22890    23849    24647.5 25327    25832.5 26130    26237
26117.5 25766    25226.5 24509.5 23675    22860    22161    21531    20934
20404.5 20015.5 19619    19396.5 19309    19296    19082    18705    18660
19259    19385.5 19098    19093    19317.5 19446    19286.5 19275.5 19199
19171.5 18758];
    elseif i==4
        expdata=[0 78.294    584.5    1295.5    2058.9    2794.3    3467.05
4099.4    4652.75 5184.15 5693.65 6131.4    6549.8    6960.3    7353.75 7697.25
8050.75 8371.05 8697.05 8987.3    9279.05 9581.5    9843.1    10101.55
10413    10876.5 11622    12405    13143    13891.5 14622    15196.5 15731.5
16098    16360.5 16433    16367    16110    15742.5 15233.5 14640.5 14039
13493.5 12955.5 12382    12081.5 11936    11877.5 11958.5 11980.5 11993
11977    11949.5 11858.5 11969.5 11983    11825.5 11698    11571    11515
11315    11197];
    end
    maxpoints_exp(i)=max(expdata'); %max viscosity of overshoot
end

%minimize error between model and experiment
[scale_mb2, fval,
exitflag]=fmincon(@MB2,f0,[],[],[],[],[1],[4],[],options);
re_mb2=53/scale_mb2; %effective aspect ratio
scale_mb2; %scaling factor

function e=MB2(hscale)
%this sub-function calculates error between model and experiment
for
    %fmincon so fmincon can optimize the aspect ratio scaling factor
    orient=[.586 0 .207]; %experimental initial orientation
    (a11,a12,a22)
    h=53/hscale;
    for j=1:4
        %calculate model prediction
        [t,eta,div] =
shear_pde_solver(rate(j),wt,sig(1),CI(1),etap,lambda,alpha,tspan,modes,
h,orient);
        maxpoints(j)=max(eta); %max viscosity point in shear
overshoot in model prediction
    end
    r0=0.01;
    %Linear viscoelastic plateau: used data for trouton at
0.01s^-1 (3*eta) between 1 and 100 sec

```

```

        time_trout=[0 1.125 1.28 1.455 1.655 1.885 2.145
2.44 2.775 3.155 3.59 4.085 4.645 5.285 6.015 6.845
7.79 8.865 10.085 11.475 13.06 14.865 16.915 19.245 21.9
24.92 28.355 32.265 36.715 41.78 47.545 54.105 61.565 70.055
79.72 90.72];
        exp_trout=[0 83514 88821 93177 98445 103725 108942
113946 118962 124137 129135 134151 139248 144234 149094 153879
159309 164502 168870 173403 177450 181362 184707 187107 190683
195153 198306 200370 202899 204825 205572 207951 208434 211290
212115 212574];
        orient1=[.586 .207 .207]; %experimental initial orientation
(a11,a22,a33)

        %calculate model prediction

[t,ext,div]=extensional_pde_solver(r0,wt,sig(1),CI(1),etap,lambda,alpha
,time_trout,modes,h,orient1);
        a=find(t>1 & t<100); %model prediction for linear
viscoelastic plateau
        e_trout=0; %initialize error variable
        %calculate error of model prediction of LVE plateau
        for k=2:length(a)
            e_trout=e_trout+(log10(exp_trout(k))-
log10(ext(a(k))))^2;
        end
        %calculate total error for mb2
        e=(sum((log10(maxpoints_exp)-log10(maxpoints)).^2))+z*e_trout;
    end

%MB2-HHT
for i=1:4
    %experimental viscosity
    if i==1
        expdata=[0 47.56 606.43 1449.7 2348.5 3178.3 3978.3
4677.2 5311.3 5910.9 6447.8 6956.1 7433.9 7911 8342.7 8777.3
9160.9 9561.8 9965.6 10310 10633 10989 11310 11617 11936
12170 12487 12903 13640 14766 15862 16922 18042 19203
20436 21708 23068 24427 25852 27324 28686 30031 31401
32729 34119 35448 36699 38064 39314 40295 41652 42824
43781 44732 45417 46413 47050 47539 48099 48349 48675
48741 48758 48700 48503 48247 47970 47606 47242 46890
46533 46219 45951 45749 45587 45514 45503 45427 45343
45249 45145 45181 45302 45257 45066 44909 44915 44746
44712 44278 44189 43827 43584 43794 43760 43936];
    elseif i==2
        expdata=[0 66.031 624.41 1415 2256.7 3048.4 3757.8
4418.8 5029.6 5591.8 6115.1 6603.6 7075.5 7510.8 7905.8 8311
8684.8 9049.8 9405.8 9731.3 10069 10373 10663 10960 11241
11512 11779 12170 12897 13781 14677 15656 16606 17564
18662 19842 21006 22105 23166 24286 25339 26509 27518
28569 29665 30583 31487 32281 33070 33657 34197 34532
34771 34785 34647 34332 33770 33081 32301 31465 30681
30001 29422 28873 28377 27963 27606 27475 27409 27087
26845 26816 26650 26670 26531 26342 25856 25536 25542
25261 25432 25555 25623 25484];
    elseif i==3

```

```

expdata=[0 87.354 628.09 1384.5 2182.5 2935 3619.7
4239.6 4826.3 5366.2 5836.6 6320.1 6761.8 7166.7 7542.3 7928.6
8241.4 8592.8 8879.2 9196.6 9512.9 9765.7 10042 10323 10559
10822 11075 11311 11747 12442 13335 14168 14970 15923
16845 17730 18594 19405 20269 21011 21711 22367 22907
23391 23716 23927 24009 23952 23764 23433 23011 22523
22006 21492 20996 20534 20078 19625 19248 18957 18707
18433 18248 18210 18207 18220 18286 18058 17770 17614
17452 17137 16801 16329 15954 15278 15025 14791 14257];
elseif i==4
expdata=[0 82.036 606.86 1335.9 2110.6 2844.7 3506.9
4095.5 4625.8 5119.7 5589.3 6035.9 6411.6 6741.9 7117.2 7440.7
7745.2 8032.4 8299.3 8550.5 8808.2 9026.2 9261.2 9544 9730.8
9949.6 10173 10344 10786 11417 12030 12639 13152 13567
13980 14230 14400 14420 14353 14139 13851 13457 13004
12563 12119 11723 11333 11006 10731 10514 10373 10274
10158 10107 10166 10208 10270 10335 10306 10145 10120
10129 10067 10013 9953.8 9964.8 10036];
end
maxpoints_exp(i)=max(expdata'); %max viscosity of overshoot
end

%minimize error between model and experiment
[scale_mb2hht, fval,
exitflag]=fmincon(@MB2HHT,f0,[],[],[],[],[1],[4],[],options);
re_mb2hht=44/scale_mb2hht; %effective aspect ratio
scale_mb2hht; %scaling factor

function e=MB2HHT(hscale)
%this sub-function calculates error between model and experiment
for
%fmincon so fmincon can optimize the aspect ratio scaling factor
orient=[.441 0 .2795]; %experimental initial orientation
(a11,a12,a22)
h=44/hscale;
for j=1:4
%calculate model prediction
[t,eta,div] =
shear_pde_solver(rate(j),wt,sig(2),CI(2),etap,lambda,alpha,tspan,modes,
h,orient);
maxpoints(j)=max(eta); %max viscosity point in shear
overshoot in model prediction
end
r0=0.01;
%Linear viscoelastic plateau: used data from 0.0001s^-1
(n*eta, n=3.8) between 1 and 100 sec
time_trout=[0 1.075 1.22 1.385 1.575 1.79 2.035
2.31 2.625 2.985 3.39 3.85 4.375 4.97 5.645 6.415
7.29 8.285 9.415 10.7 12.16 13.82 15.705 17.845 20.28
23.05 26.195 29.77 33.835 38.45 43.7 49.665 56.44 64.145
72.9 82.845 94.15];
exp_trout=[0 113661.8 99913.4 119114.8 119525.2
116078.6 132342.6 131882.8 152311.6 159824.2 146053
163673.6 164574.2 164619.8 177631 180370.8 184212.6
181678 193792.4 197045.2 197980 212610 213875.4 218294.8

```



```

216182 219548.8 242896 232172.4 230553.6 233377 240585.6
244062.6 238058.6 243112.6 250689.8 252787.4 245324.2];
orient1=[.441 .2795 .2795]; %experimental initial
orientation (a11,a22,a33)

%calculate model prediction

[t,ext,div]=extensional_pde_solver(r0,wt,sig(2),CI(2),etap,lambda,alpha
,time_trout,modes,h,orient1);
a=find(t>1 & t<100); %model prediction for linear
viscoelastic plateau
e_trout=0; %initialize error variable
%calculate error of model prediction of LVE plateau
for k=2:length(a)
    e_trout=e_trout+(log10(exp_trout(k))-
log10(ext(a(k))))^2;
end
%calculate total error for mb2hht
e=(sum((log10(maxpoints_exp)-log10(maxpoints)).^2))+z*e_trout;
end

%SC2
for i=1:4
    %experimental viscosity
    if i==1
        expdata=[0 46.69 534.64 1332.6 2176.3 2964.8 3727.8
4412.9 5013.3 5591.9 6170.2 6691.2 7173.6 7642.8 8091.4 8493.5
8883.1 9289.5 9642.4 9972 10353 10704 11007 11334 11646
11934 12221 12630 13447 14563 15682 16835 17961 19139
20369 21656 22966 24300 25739 27163 28570 30014 31453
32987 34555 35976 37305 38609 40282 41632 42792 43995
45096 45865 46987 47621 48093 48491 48618 48453 48145
47535 46686 45747 44626 43371 42111 40947 40067 39452
38987 38733 38722 38947 39274 39675 39927 40047 40079
40401 40765 40633 40822 40897 41109 41582 41733 41999
42335 42222 42799 42591 42464 42640 43358 44800];
    elseif i==2
        expdata=[0 1.7318 546.57 1358.4 2224.9 3051.1 3824.1
4527.7 5180 5776.1 6333.1 6842.5 7329.9 7808.9 8244.6 8681.6
9047.8 9481.8 9860.3 10224 10572 10907 11203 11517 11832
12153 12453 12874 13704 14730 15835 17078 18362 19554
20856 22145 23421 24779 26081 27376 28694 29895 31154
32385 33357 34290 35010 35745 36162 36356 36445 36218
35860 35244 34470 33662 32841 32027 31258 30592 30145
29850 29780 29898 30123 30304 30329 30249 30213 30149
29944 29640 29198 28897 28697 28995 29499 30168 30387
29750 29243 29098 30009 30418 30073 30110 30002 30242
29961 30404 31371];
    elseif i==3
        expdata=[0 83.562 635.88 1426 2283 3094.4 3843.7
4523.4 5155.8 5733.9 6312.9 6823.6 7291.8 7762 8188 8612.7
9032.3 9384.7 9755.5 10029 10417 10735 11029 11349 11635
11933 12268 12515 13071 13947 14939 15939 16941 17989
18969 19988 20965 21849 22637 23360 23937 24322 24533
24502 24247 23711 22939 21997 20968 19850 18720 17770
17262 17162 17303 17616 18041 18401 18610 18591 18525

```

```

18315    18014    17718    17554    17912    18391    18547    18259    18296
18344    18484    18327    18469    18667    18545    18048    17298    17227
16680    15784    14849    14048    14042    14292    13739    13411    13484
14157];
elseif i==4
    expdata=[0 71.359    585.85    1288.8    2060.1    2807    3484
4112.1    4690.9    5193.1    5668.5    6120.5    6540.9    6912.2    7291.7    7589.7
7944.7    8231.9    8528.3    8813.8    9118.9    9353.9    9592    9839.8    10067
10278    10477    10665    11086    11760    12342    12867    13415    13797
14099    14302    14417    14446    14352    14110    13858    13506    13146
12793    12486    12169    11887    11663    11463    11275    11086    10901
10753    10580    10532    10572    10461    10248    10124    10108    10275
9986.5    9727.1    9464.9    9189.8    9304.7    9346.9    9218.6    8557.1    7401.4
6479    6499.6    6478.8    6392.4    6535.5    6508.1    6727.7    6844.7    6884.1
6761.7    6629.9    6373.8    6136.3];
end
maxpoints_exp(i)=max(expdata'); %max viscosity of overshoot
end

%minimize error between model and experiment
[scale_sc2, fval,
exitflag]=fmincon(@SC2,f0,[],[],[],[],[1],[4],[],options);
re_SC2=69/scale_sc2; %effective aspect ratio
scale_sc2; %scaling factor

function e=SC2(hscale)
%this sub-function calculates error between model and experiment
for
    %fmincon so fmincon can optimize the aspect ratio scaling factor
    orient=[.528 0 .236]; %experimental initial orientation
    (a11,a12,a22)
    h=69/hscale;
    for j=1:4
        %calculate model prediction
        [t,eta,div] =
shear_pde_solver(rate(j),wt,sig(3),CI(3),etap,lambda,alpha,tspan,modes,
h,orient);
        maxpoints(j)=max(eta); %max viscosity point in shear
overshoot in model prediction
    end
    r0=0.01;
    %Linear viscoelastic plateau: used data from 0.001s^-1
    (n*eta, n=5.5) between 1 and 100 sec
    time_trout=[0 1.01    1.15    1.31    1.49    1.695    1.93
2.195    2.5    2.85    3.245    3.69    4.2    4.785    5.445    6.2    7.06
8.035    9.15    10.42    11.865    13.51    15.38    17.51    19.935    22.7
25.85    29.43    33.51    38.155    43.445    49.47    56.325    64.13    73.02
83.145    94.675];
    exp_trout=[0 150920 160611 171561.5    181549.5
191889.5    199160.5    210743.5    222717    233722.5    241125.5
254578.5    265589.5    274334.5    284971.5    296015.5    304942
314418.5    323306.5    333388    341660    349266.5    356438.5    363352
370507.5    376623.5    382299.5    386996.5    391968.5    396995.5
401362.5    404827.5    407539    410217.5    413693.5    416152
421635.5];

```

```

orient1=[.528 .236 .236]; %experimental initial orientation
(a11,a22,a33)

%calculate model prediction

[t,ext,div]=extensional_pde_solver(r0,wt,sig(3),CI(3),etap,lambda,alpha
,time_trout,modes,h,orient1);
a=find(t>1 & t<100); %model prediction for linear
viscoelastic plateau
e_trout=0; %initialize error variable
%calculate error of model prediction of LVE plateau
for k=2:length(a)
    e_trout=e_trout+(log10(exp_trout(k))-
log10(ext(a(k))))^2;
end
%calculate total error for sc2
e=(sum((log10(maxpoints_exp)-log10(maxpoints)).^2))+z*e_trout;
end

%display optimized aspect ratio scaling factors
scale_mb2
scale_mb2hht
scale_sc2

end

```

Composite Parameter Optimization: σ

```
function sigma_optimization
%This function optimizes sigma for MB2,MB2HHT, and SC2 for given values
of
%of the aspect ratio scaling factors. These results can then be used
%in "aspect_ratio_optimization" to optimize aspect ratio scaling facotr
for
%these sigmas.
%Iteratively use these two programs until the values for the scaling
%factors and sigmas converge

hscale=[1.8443 1.4487 1.7566]; %change with iterations with
"aspect_ratio_optimization"

CI=[.0306,.0301,.0499]; %optimized values from "ci_optimization"
program

%steady-state values for shear viscosity
ss_mb2=[46446.68182 33019.95 19162.46429 11904.21429]; %averaged from
t=[103,400] for r=0.1, t=[34,105] for r=0.3, t=[19,111] for r=1,
t=[4,25] for r=3
ss_mb2hht=[44760.23529 26285.09091 18196.2 10099.5875]; %averaged from
t=[177,1391] for r=0.1, t=[57,219] for r=0.3, t=[23,39] for r=1,
t=[6,41] for r=3
ss_sc2=[39842.375 29802.24 18115 10362.5]; %averaged from t=[46,337]
for r=0.1, t=[18,417] for r=0.3, t=[21,57] for r=1, t=[7,17] for r=3

%optimized from pure polymer
etap=[1563.180023 10503.23358 30793.85272 9864.433234 9310.030642
40690.01669];
lambda=[0.01461114 0.235979496 3.210427777 34.39722848 6390.005373
116255.5053];
alpha=[0.818868357 0.56 0.56 0.078601046 0.025189799
0.001841609];
rate=[.1,.3,1,3]; %shear rates
wt=.02; %mass fraction of cnfs
modes=6;
tspan=0:100;

f0=.5; %initial guess for fmincon
options=optimset('Algorithm','interior-point'); %search algorithm

%MB2
[sigma_mb2, fval,
exitflag]=fmincon(@MB2,f0,[],[],[],[],[.5],[1],[],options);

function e=MB2(sigm)
%this sub-function calculates error between model and experiment
for
%fmincon so fmincon can optimize sigma
orient=[.586 0 .207]; %experimental orientation (a11,a12,a22)
h=53/hscale(1); %effective aspect ratio
ss_visc=zeros(1,4);
for j=1:4
```

```

        %calculate model prediction
        [t,eta] =
shear_pde_solver(rate(j),wt,sigm,CI(1),etap,lambda,alpha,tspan,modes,h,
orient);
        ss_visc(j)=eta(end); %find model prediction for steady
state viscosity
    end
    e=sum((ss_mb2-ss_visc).^2); %total error from all shear rates
end

%MB2HHT
[sigma_mb2hht, fval,
exitflag]=fmincon(@MB2HHT,f0,[],[],[],[],[.5],[1],[],options);

function e=MB2HHT(sigm)
%this sub-function calculates error between model and experiment
for
%fmincon so fmincon can optimize sigma
    orient=[.441 0 .2795]; %experimental orientation (a11,a12,a22)
    h=44/hscale(2); %effective aspect ratio
    ss_visc=zeros(1,4);
    for j=1:4
        %calculate model prediction
        [t,eta] =
shear_pde_solver(rate(j),wt,sigm,CI(2),etap,lambda,alpha,tspan,modes,h,
orient);
        ss_visc(j)=eta(end); %find model prediction for steady
state viscosity
    end
    e=sum((ss_mb2hht-ss_visc).^2); %total error from all shear
rates
end

%SC2
[sigma_sc2, fval,
exitflag]=fmincon(@SC2,f0,[],[],[],[],[.5],[1],[],options);

function e=SC2(sigm)
%this sub-function calculates error between model and experiment
for
%fmincon so fmincon can optimize sigma
    orient=[.528 0 .236]; %experimental orientation (a11,a12,a22)
    h=69/hscale(3); %effective aspect ratio
    ss_visc=zeros(1,4);
    for j=1:4
        %calculate model prediction
        [t,eta] =
shear_pde_solver(rate(j),wt,sigm,CI(3),etap,lambda,alpha,tspan,modes,h,
orient);
        ss_visc(j)=eta(end); %find model prediction for steady
state viscosity
    end
    e=sum((ss_sc2-ss_visc).^2); %total error from all shear rates
end

%display optimized values

```

```
sigma_mb2  
sigma_mb2hht  
sigma_sc2
```

```
end
```

Composite Model Predictions: Melt Blended with O-CNFS (MB)

```
function MB_2wt
%This program plots the viscosity for the MB 2wt% composite for
transient
%extension and transient shear as well as moduli predictions for SAOS
% given the optimized values for sigma and CI and hfactor

format long g
sigma=.809;
CI=.0306;
hfactor=1.84;

para=[ 1563.180023    10503.23358 30793.85272 9864.433234 9310.030642
40690.01669; %optimized values for etap by mode
      0.01461114    0.235979496 3.210427777 34.39722848 6390.005373
116255.5053; %optimized values for lambda by mode
      0.818868357 0.56          0.56          0.078601046 0.025189799
0.001841609]; %optimized values for alpha by mode
modes=6;
wt=.02; % mass fraction of cnfs
re=53/hfactor; %53= number ave length, 74= weight ave length
orient_ext=[.586, .207, .207]; %experimentally determined initial
orientation of fibers (a11,a22,a33)
orient_shr=[.586 0 .207]; %%experimentally determined initial
orientation of fibers (a11,a12,a22)

%%
%%Part 1: Extensional plotting
terr=zeros(1,5); %error per extension rate in model predictions
tdiv=0; %keeps track of any divergences in extensional model
predictions

for m=1:5 % m designates the m-th extension rate
    if m==1
        r=.01;
        %truncated, set .01(3), time and viscosity
        timedata=[0 31.90831    38.20769    45.7507 54.78287
65.59817    78.54865    94.05583    112.6245    134.8589    161.483
193.3631    231.5372    277.2475    331.9821    397.5225    476.0019
569.9748    682.5]';
        expdata=[0 289168.7    288428.7    292556.6    294624.6
310069.7    320538.6    330746.3    343389.5    360977.1    406048.1
448489.1    530888.7    671355.9    919199.6    1431823    2544252    5630486
23059090]';
    elseif m==2
        r=.03;
        %truncated, set .03(2)
        timedata=[0 11.35781    13.29856    15.57092    18.23158
21.34686    24.99447    29.26535    34.26601    40.12115    46.97677
55.00384    64.40251    75.40717    88.29222    103.379    121.0437
141.7268    165.9441    194.2995    227.5]';
        expdata=[0 242964.2    249710.6    259360.2    265015.3
284672.8    297642.1    307550.8    325966.1    339455.4    349054.5
```

```

382981.4    422765.9    469610.8    565065.9    764341.5    1133653
1886361 3582319 7532099 14584450]';
elseif m==3
    r=.1;
    %set .1(4)
    timedata=[0 0.1 0.1142454 0.1305202 0.1491133
0.1703552 0.194623 0.2223479 0.2540223 0.2902088 0.3315503
0.3787811 0.4327401 0.4943858 0.5648131 0.6452732 0.7371951
0.8422117 0.9621884 1.099256 1.25585 1.434751 1.639138
1.87264 2.139405 2.444173 2.792356 3.190139 3.644588
4.163775 4.756922 5.434566 6.208744 7.093206 8.103663
9.258065 10.57692 12.08364 13.80501 15.77159 18.01832
20.58511 23.51755 26.86772 30.69514 35.0678 40.06336
45.77055 52.29077 59.73981 68.25]';
    expdata=[0 6890.88 16548.34 19710.05 32912.64
44620.99 45268.02 49846.2 55173.66 60234.97 63974.17
75411.8 77752.9 79499.09 84612.06 93833.12 109072.4
108508.3 124097.4 129830.1 144399.7 152719.9 167945.9
163324.8 174523.3 151088.1 155467.1 158789.5 165107.9
176287.1 179696.5 188722.5 196268.9 206751.6 216295
223766.6 234607.3 243259.2 257513.2 263149.5 265260.4
278204.8 298160.7 331136.5 384562.6 529194.5 590804.1
684427.5 855833.6 1237321 2425558]';
elseif m==4
    r=.3;
    %set .3(2)
    timedata=[0 0.1 0.1117125 0.1247968 0.1394135
0.1557423 0.1739836 0.1943614 0.2171259 0.2425567 0.2709661
0.3027029 0.3381569 0.3777634 0.4220089 0.4714366 0.5266534
0.5883376 0.6572464 0.7342262 0.8202223 0.9162906 1.023611
1.143501 1.277433 1.427052 1.594195 1.780915 1.989504
2.222524 2.482837 2.773638 3.0985 3.461411 3.866828
4.319729 4.825676 5.390882 6.022287 6.727646 7.51562
8.395884 9.37925 10.47779 11.705 13.07595 14.60746
16.31836 18.22964 20.36478 22.75]';
    expdata=[0 15244.8 16808.13 19480.84 24398.25
31463.05 34981.9 35769.67 43030.57 48934.49 48372.1
56862.49 59324.29 66840.18 71717.51 77075.24 79760.8
86767.66 93515.34 100470.5 102423.9 104845.2 99072.72
102442 107144.2 112945.6 119091 124251.3 130352.7
136666.7 142334.2 148959.1 156724.6 165724.5 174678.5
181392.2 189530.8 194037.3 204958.5 220579.5 237540.7
258784.2 297150.3 356699.3 410071.4 444201.3 493379.5
543629.5 541386.1 695399.3 1190882]';
elseif m==5
    r=1;
    %set 1(3)
    timedata=[0 0.1 0.109001 0.1188123 0.1295066
0.1411636 0.1538698 0.1677197 0.1828162 0.1992716 0.2172081
0.2367591 0.2580699 0.2812989 0.3066187 0.3342176 0.3643007
0.3970916 0.432834 0.4717935 0.5142599 0.5605487 0.6110039
0.6660007 0.7259477 0.7912906 0.862515 0.9401504 1.024774
1.117014 1.217557 1.32715 1.446607 1.576817 1.718747
1.873452 2.042083 2.225892 2.426245 2.644632 2.882677
3.142148 3.424974 3.733258 4.06929]';
    expdata=[0 11681.41 13905.72 16483.72 19666.6
23990.51 28958.02 31093.06 33655.69 37375.28 41634.22

```



```

43956.18    46473.75    51156.52    54938.75    54887.86    55067.26
57913.16    61620.79    65169.65    69197.97    73716.18    78080.86
82602.3 87552.7 93346.01    99427.7 106027.7    112532.7    119715.9
127646.6    135435.2    143607.9    151167.4    159694.1    171481.5
184090.3    199013.9    219175    240814.2    268412.1    302961.1
341944.7    384204.2    409236.3]';
    end

    %differential equation solver to get model prediction of viscosity
    vs time

    [t,etatep,diverge]=extensional_pde_solver(r,wt,sigma,CI,para(1,:),para
    (2,:),para(3,:),timedata,modes,re,orient_ext);

    if diverge==1 %keep track of divergence
        tdiv=1;
        terr(m)=inf;
    elseif diverge==0 %if no divergence, calculate error between model
    and experiment
        for k=2:length(timedata)
            terr(m)=terr(m)+(log10(expdata(k))-log10(etatep(k)))^2;
        end
    end
end

%plot the model predictions and experimental data
if m==1
    figure
    if diverge==0
        loglog(t,etatep,'r',timedata,expdata,'*r');
    end
    hold on
elseif m==2
    if diverge==0
        loglog(t,etatep,'g',timedata,expdata,'*g');
    end
elseif m==3
    if diverge==0
        loglog(t,etatep,'b',timedata,expdata,'*b');
    end
elseif m==4
    if diverge==0
        loglog(t,etatep,'c',timedata,expdata,'*c');
    end
elseif m==5
    if diverge==0
        loglog(t,etatep,'k',timedata,expdata,'*k');
    end

    title('Extensional Viscosity MB 2wt%');
    xlabel('Time (s)');
    ylabel('Viscosity (Pa*s)');
    legend('Mod \epsilon=.01','Exp \epsilon=.01','Mod
\epsilon=.03','Exp \epsilon=.03','Mod \epsilon=.1','Exp
\epsilon=.1','Mod \epsilon=.3','Exp \epsilon=.3','Mod \epsilon=1','Exp
\epsilon=1',-1);

```

```

        set(gcf,'Units','normalized',
'WindowStyle','docked','OuterPosition',[0 0 1 1]); %this places the
plot in a maximized window
        hold off
    end

end

Error=sum(terr); %sum of error from all extension rates

%%
%%Part 2: Shear plotting
serr=zeros(1,5); %error per shear rate in model predictions
sdiv=0; %keeps track of any divergences in shear model predictions

for w=1:5 % w designates the w-th shear rate
    if w==1
        r=.01;
        %set .01(2)
        timedata=[0 0.01 0.02 0.03 0.04 0.05 0.06
0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15
0.16 0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.24
0.25 0.26 0.27 0.28 0.29 0.3 0.31 0.32 0.33
0.34 0.35 0.36 0.37 0.38 0.39 0.4 0.41 0.42
0.43 0.44 0.45 0.46 0.47 0.48 0.49 0.5 0.51
0.52 0.53 0.54 0.55 0.56 0.57 0.58 0.59 0.6
0.61 0.62 0.65 0.7 0.75 0.8 0.855 0.915 0.975 1.04
1.115 1.195 1.28 1.37 1.465 1.565 1.675 1.795 1.92
2.05 2.19 2.345 2.51 2.685 2.875 3.08 3.295 3.525
3.77 4.03 4.315 4.62 4.94 5.285 5.655 6.05 6.475
6.93 7.42 7.94 8.495 9.09 9.725 10.41 11.14 11.92
12.755 13.65 14.61 15.63 16.725 17.9 19.155 20.495 21.93
23.47 25.115 26.875 28.76 30.775 32.935 35.245 37.715 40.36
43.19 46.22 49.46 52.925 56.635 60.61 64.86 69.405 74.27
79.48 85.055 91.02 97.4 104.22 111.53 119.36 127.73 136.68
146.27 156.52 167.5 179.24 191.82 205.26 219.65 235.06 251.54
269.18 288.06 318.7 341.04 364.96 390.55 417.94 447.24 478.61
512.17 548.08 586.52 627.64 671.66 718.76 769.16 823.09 880.82
942.58 1008.7 1079.4 1155.1 1236.1 1322.8 1415.5 1514.8 1621
1734.7 1856.3 1986.5 2125.8 2274.9 2434.4 2605.1 2787.8 2983.3
3192.5 3416.3 3655.9];
        expdata=[0 86.632 490.45 1141.1 1980.1 2802.5 3502.8
4159.4 4967 5516.4 5839.9 6362.8 6919.1 7355.4 7592.2 8024.5
8481.5 8837.5 9211.8 9629.1 9869.9 10189 10446 10878 11089
11333 11695 12074 12278 12396 12533 13057 13212 13304
13763 14040 14209 14305 14672 14909 14988 15419 15779
15709 15877 16202 16524 16356 16660 16979 17076 17326
17524 17784 17606 17773 18226 18183 18104 18535 18824
18837 18892 19368 20036 20711 21354 21923 22707 23363
24137 25027 25783 26572 27449 28141 28936 29905 30604
31397 32191 32995 33670 34543 35557 36510 37229 38016
38955 39986 40701 41471 42450 43468 44362 45178 45816
46413 47332 47849 48622 49248 49922 50602 51325 51898
52544 53100 53770 54327 54860 55603 56233 56791 57426
57845 58002 58251 58869 59389 59760 60072 60345 60468
60798 61485 62214 62142 62327 62721 62998 62871 63217

```

```

63306    63167    63867    63902    64177    63997    64257    64163    64840
64860    64796    64692    65239    65143    65213    65391    65445    65393
65843    65655    65766    65715    65890    65737    65947    65887    65831
65894    65992    66072    66268    66306    66341    66285    66326    66606
66494    66813    66754    66819    67025    67161    66853    67071    67055
66968    66602    66396    66315    65891    65984    66252    66269    66709
67091    66863    66350    65831];
elseif w==2
    r=.1;
    %set ave of 1 and 2
    timedata=[0 0.01 0.02 0.03 0.04 0.05 0.06
0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15
0.16 0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.24
0.25 0.26 0.28 0.315 0.355 0.4 0.455 0.52 0.595
0.675 0.765 0.87 0.99 1.125 1.28 1.455 1.655 1.885
2.145 2.44 2.775 3.155 3.59 4.085 4.645 5.285 6.015
6.845 7.79 8.865 10.085 11.475 13.06 14.865 16.915 19.245
21.9 24.92 28.355 32.265 36.715 41.78 47.545 54.105 61.565
70.055 79.72 90.72 103.24 117.47 133.68 152.12 184.27 209.68
238.61 271.54 308.99 351.61 400.13 455.32 518.14 589.61 670.95
763.51 868.83 988.7 1125.1 1280.3 1456.9 1657.9 1886.6 2146.9
2443 2780.1 3163.6 3600];
    expdata=[0 50.7945 584.48 1344.55 2156.55 2936.75 3642.3
4300.35 4916.1 5486.55 6033.9 6547.35 6989.2 7478.75 7894.65 8337.05
8711.15 9078.3 9467.1 9817.15 10128.55 10489.5 10776 11133.5
11398 11718.5 11999 12539.5 13462 14433 15452 16596 17857
19195.5 20482 21804.5 23187.5 24604 26029 27489 28942.5 30349
31871 33540 35291 36825.5 38329 39883.5 41336.5 42792 44163.5
45455.5 46928 48050 49140.5 50051.5 50998.5 51635.5 52298 52613.5
52941.5 52956 52840 52555 52109 51529 50807.5 50015 49252.5
48562.5 47945.5 47447.5 47096 46859.5 46687.5 46553 46500.5 46521.5
46502 46310 46213 46145 46097.5 46524 46700 46803 46926
46665 46758.5 46678.5 46505.5 45537.5 44098 43581 41589.5 40157.5
38101.5 36936.5 34339 32735 30810];
elseif w==3
    r=.3;
    %set ave 1,2,3,4
    timedata=[0 0.01 0.02 0.03 0.04 0.05 0.06
0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15
0.16 0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.24
0.255 0.285 0.325 0.37 0.425 0.49 0.56 0.64 0.735
0.84 0.96 1.1 1.26 1.44 1.645 1.885 2.16 2.475
2.835 3.245 3.715 4.255 4.875 5.585 6.4 7.33 8.395
9.62 11.02 12.625 14.465 16.57 18.98 21.74 24.905 28.535
32.69 37.45 42.905 49.15 56.31 64.51 73.905 84.67 97
111.13 127.31 145.85 178.43 204.41];
    expdata=[0 99.9685 604.9975 1340.875 2132.75
2901.175 3615.725 4250.8 4849.225 5396.025 5902.575
6368.925 6840.575 7264.625 7669.825 8073.225 8450.65
8810.425 9167.825 9500.225 9835.875 10147.425 10466.975
10761.575 11107.75 11687.75 12541.75 13574.25 14676
15808 17049 18279.75 19525.75 20877.5 22255 23687
25110.75 26524 27946.75 29342.75 30698 32143.5 33463.25
34690.75 35815.5 36905.75 37777 38536.75 39114.5 39512
39696.75 39695 39474.25 39051.75 38471 37797 37087.5
36418.25 35882 35513.25 35195.25 34831.25 34404.5 34004.5

```

```

33589.5 33141.75 32648.25 32318.75 32091.5 31819 31350.5
30692 30481.25 30675.25 30402.25];
elseif w==4
    r=1;
    %set ave 1,2 truncated
    timedata=[0 0.01 0.02 0.03 0.04 0.05 0.06
0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15
0.16 0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.245
0.27 0.305 0.35 0.405 0.465 0.535 0.615 0.705 0.81
0.93 1.07 1.23 1.41 1.62 1.86 2.135 2.455 2.82
3.24 3.725 4.285 4.925 5.66 6.505 7.475 8.59 9.87
11.345 13.04 14.985 17.225 19.8 22.755 26.15 30.055 34.545
39.705 45.635 52.45 60.285 69.295 79.65 91.545 105.22 120.94
139];
    expdata=[0 75.508 584.995 1305.55 2089.8 2826.1 3530.25
4155.5 4741.3 5265.55 5766.4 6218.95 6653.3 7066.3 7452.2 7830.9
8208.15 8569.3 8903.4 9235.3 9574 9888.8 10211.55 10576.5
11194 12001.5 12934.5 13984 15073 16183 17332 18511 19647
20766.5 21847.5 22890 23849 24647.5 25327 25832.5 26130 26237
26117.5 25766 25226.5 24509.5 23675 22860 22161 21531 20934
20404.5 20015.5 19619 19396.5 19309 19296 19082 18705 18660
19259 19385.5 19098 19093 19317.5 19446 19286.5 19275.5 19199
19171.5 18758];
elseif w==5
    r=3;
    %set ave 1,2 truncated
    timedata=[0 0.01 0.02 0.03 0.04 0.05 0.06
0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15
0.16 0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.245
0.27 0.31 0.355 0.405 0.465 0.535 0.615 0.705 0.81
0.93 1.07 1.23 1.415 1.625 1.865 2.145 2.465 2.83
3.255 3.745 4.3 4.94 5.68 6.53 7.505 8.625 9.91
11.39 13.095 15.05 17.3 19.885 22.855 26.27 30.195 34.71
39.9];
    expdata=[0 78.294 584.5 1295.5 2058.9 2794.3 3467.05
4099.4 4652.75 5184.15 5693.65 6131.4 6549.8 6960.3 7353.75 7697.25
8050.75 8371.05 8697.05 8987.3 9279.05 9581.5 9843.1 10101.55
10413 10876.5 11622 12405 13143 13891.5 14622 15196.5 15731.5
16098 16360.5 16433 16367 16110 15742.5 15233.5 14640.5 14039
13493.5 12955.5 12382 12081.5 11936 11877.5 11958.5 11980.5 11993
11977 11949.5 11858.5 11969.5 11983 11825.5 11698 11571 11515
11315 11197];
end
%shift time data back to account for start up delay in
rheometer
factor=timedata(2)-0.0005;
timedata=timedata-factor;
timedata(1)=0;

%differential equation solver to get model prediction of
viscosity vs time

[t,etatemp,diverge]=shear_pde_solver(r,wt,sigma,CI,para(1,:),para(2,:),
para(3,:),timedata,modes,re,orient_shr);

if diverge==1 %keep track of divergence

```

```

        sdiv=1;
        serr(w)=inf;
    elseif diverge==0 %if no divergence, calculate error between
model and experiment
        for k=2:length(timedata)
            serr(w)=serr(w)+(log10(expdata(k))-
log10(etatemp(k)))^2;
        end
    end

    %plot the model predictions and experimental data
    if w==1
        figure
        if diverge==0
            loglog(timedata,expdata,'*',t,etatemp,'Color',[0,0,1]);
        end
        hold on
    elseif w==2
        if diverge==0
            loglog(timedata,expdata,'*',t,etatemp,'Color',[0,0,0]);
        end
    elseif w==3
        if diverge==0
            loglog(timedata,expdata,'*',t,etatemp,'Color',[0,1,1]);
        end
    elseif w==4
        if diverge==0
            loglog(timedata,expdata,'*',t,etatemp,'Color',[.6,0,.9]);
        end
    elseif w==5
        if diverge==0
            loglog(timedata,expdata,'*',t,etatemp,'Color',[.7,.9,.3]);
        end

        xlabel('Time (s)')
        ylabel('Viscosity (Pa*s)')
        title('Shear Viscosity MB 2wt%')
        legend('Exp \gamma=0.01','Mod \gamma=0.01','Exp
\gamma=0.1','Mod \gamma=0.1','Exp \gamma=.3','Mod \gamma=.3','Exp
\gamma=1','Mod \gamma=1','Exp \gamma=3','Mod \gamma=3',-1);
        set(gcf,'Units','normalized',
'WindowStyle','docked','OuterPosition',[0 0 1 1]);
        hold off
    end
end
Error=sum(serr); %sum of error from all shear rates

%%
%%PART 3: SAOS plotting
%read in sigma and CI values to be sent to other programs
%call SAOSModeling variant to calc experimental stress.
%call SAOSFittingGpGdp variant to calc model prediction.
%find error between the two and send back as Gerror

%NOTE: diverging probably kills the programs this subfunction calls

```

```

Gerror=0; %error in SAOS model predictions
gdiv=0; %keeps track of any divergences in SAOS model predictions

%data from Koki Data
freq=[100    63.096   39.811   25.119   15.849   10    6.3096   3.9811   2.5119
1.5849   1    0.63096  0.39811  0.25119  0.15849  0.1    0.063096    0.039811
0.025119    0.015849    0.01 ];
%storage modulus
GpExp=[130960    114830    98906    84285    70796    58502    47372    37583
29009    21859    15970    11238    7708.9    5039.8    3136.6    1919.4    1121.7
600.49    329.64    157.37    81.182];
%loss modulus
GdpExp=[63437    57164    52141    47527    43278    39018    34902    30717
26605    22503    18723    15144    11977    9210.6    6844.8    5003.9    3512.5
2444.1    1647    1089.6    704.4];

% this commented-out section calls on 'SAOSModeling' which calculates
and
% plots the stress wave from SAOS flow. Uncomment if you desire to see
% this, but only use every third point from the experimental data;
% otherwise, the computations are way too many and tedious and MATLAB
struggles
% [export] =
SAOSModeling(para,sigma,CI,freq,GpExp,GdpExp,modes,graph);
% time=export(:,1);
% taucl2=export(:,2);
% pred=export(:,3);

graph=0; %=0 for no plotting in SAOSFittingGpGdp, =1 for
plotting
%fit G' G'' of composite to SAOS stress wave
[GpGdp,diverge] =
SAOSFittingGpGdp(para,sigma,CI,freq,GpExp,GdpExp,wt,modes,graph,re,orie
nt_shr);
GpMod=GpGdp(:,1);
GdpMod=GpGdp(:,2);

if diverge==0 %if no divergence, calculate error between model
and experiment
for j=1:length(freq)
Gerror=Gerror+(log10(GpExp(j))-
log10(GpMod(j)))^2+(log10(GdpExp(j))-log10(GdpMod(j)))^2;
end
%plot model predictions and experimental data
figure
loglog(freq,GpExp,'b*',freq,GdpExp,'g*',freq,GpMod,'b',freq,GdpMod,'g')
legend('GpExp','GdpExp','GpMod','GdpMod',-1)
title('SAOS MB 2wt%')
else %keep track of divergence
Gerror=inf;
gdiv=1;
end

```

```
%%  
% uncomment to check errors and divergences if interested  
% Terror  
% Serror  
% Gerror  
% tdiv  
% sdiv  
% gdiv  
end
```

Composite Model Predictions: Melt Blended with HHT-CNFs (MBHHT)

```
function MBHHT_2wt
%This program plots the viscosity for the MB-HHT 2wt% composite for
transient
%extension and transient shear as well as moduli predictions for SAOS
% given the optimized values for sigma and CI and hfactor

format long g

sigma=.544;
CI=.0301;
hfactor=1.45;

para=[ 1563.180023 10503.23358 30793.85272 9864.433234 9310.030642
40690.01669; %optimized values for etap by mode
0.01461114 0.235979496 3.210427777 34.39722848 6390.005373
116255.5053; %optimized values for lambda by mode
0.818868357 0.56 0.56 0.078601046 0.025189799
0.001841609]; %optimized values for alpha by mode
modes=6;
wt=.02; %mass fraction of cnfs
re=44/hfactor; %44= number ave length, 70= weight ave length
orient_ext=[.441 .2795 .2795]; %experimentally determined initial
orientation of fibers (a11,a22,a33)
orient_shr=[.441 0 .2795]; %experimentally determined initial
orientation of fibers (a11,a12,a22)

%%
%%Part 1: Extensional plotting
terr=zeros(1,5); %error per extension rate in model predictions
tdiv=0; %keeps track of any divergences in extensional model
predictions

for m=1:5 % m designates the m-th extension rate
    if m==1
        r=.01;
        %set .01(2) from HHT truncated
        timedata=[0 31.90831 38.20769 45.7507 54.78287
65.59817 78.54865 94.05583 112.6245 134.8589 161.483
193.3631 231.5372 277.2475 331.9821 397.5225 476.0019
569.9748]';
        expdata=[0 210335.3 219691 226971.8 229421.2
234941.7 236315.3 242302.8 254875.7 253289.3 261135.1
267509.6 286954.2 349713.8 472989.1 757980.2 1256944
2316228]';
    elseif m==2
        r=.03;
        %set .03(2) from HHT truncated
        timedata=[0 8.284668 9.700294 11.35781 13.29856
15.57092 18.23158 21.34686 24.99447 29.26535 34.26601
40.12115 46.97677 55.00384 64.40251 75.40717 88.29222
103.379 121.0437 141.7268 165.9441 194.2995]';
```



```

expdata=[0 190341.9 190633.6 193699 196175
201475.2 205108.4 212185.2 219170.7 225653.7 230200.8
235710.5 244995.5 250231.8 248900.3 240684.8 253282.5
304421.8 390583.3 593847.9 1056209 2144129]';
elseif m==3
r=.1;
%set .1(2) from HHT truncated
timedata=[0 3.644588 4.163775 4.756922 5.434566
6.208744 7.093206 8.103663 9.258065 10.57692 12.08364
13.80501 15.77159 18.01832 20.58511 23.51755 26.86772
30.69514 35.0678 40.06336 45.77055 52.29077 59.73981
68.25]';
expdata=[0 142596.5 147838 156105.2 157534.9
165326.7 170982.4 186709.8 194717.8 199831.1 204846.9
215419.2 226120.8 237852.2 249579.6 255556 274173.5
301182.8 318106.8 354322.3 451478 668521.6 1408449
3373191]';
elseif m==4
r=.3;
%set .3(3) from HHT
timedata=[0 0.1 0.1117125 0.1247968 0.1394135
0.1557423 0.1739836 0.1943614 0.2171259 0.2425567 0.2709661
0.3027029 0.3381569 0.3777634 0.4220089 0.4714366 0.5266534
0.5883376 0.6572464 0.7342262 0.8202223 0.9162906 1.023611
1.143501 1.277433 1.427052 1.594195 1.780915 1.989504
2.222524 2.482837 2.773638 3.0985 3.461411 3.866828
4.319729 4.825676 5.390882 6.022287 6.727646 7.51562
8.395884 9.37925 10.47779 11.705 13.07595 14.60746
16.31836 18.22964 20.36478 22.75]';
expdata=[0 8633.724 11790.94 15199.42 18997.25
23847.63 27551.82 29358.34 35422.57 44593.52 42772.54
49858.95 59421.91 57570.51 53920.42 53526.64 65837.6
71856.54 79041.58 83939.48 83833.5 90149.65 96295.93
91846 94179.86 99715.19 106030.4 111699.3 117268.3
123649.4 130450.6 138892.3 146197.4 153980.6 163027.9
173915.2 184254.8 195853.7 205960.2 216452.4 240066.3
250082.8 264790.7 291294.8 322962.1 355202.6 397853.6
432315.8 493080.9 626274.2 1159989]';
elseif m==5
r=1;
%set 1(1) from HHT
timedata=[0 0.1 0.109001 0.1188123 0.1295066
0.1411636 0.1538698 0.1677197 0.1828162 0.1992716 0.2172081
0.2367591 0.2580699 0.2812989 0.3066187 0.3342176 0.3643007
0.3970916 0.432834 0.4717935 0.5142599 0.5605487 0.6110039
0.6660007 0.7259477 0.7912906 0.862515 0.9401504 1.024774
1.117014 1.217557 1.32715 1.446607 1.576817 1.718747
1.873452 2.042083 2.225892 2.426245 2.644632 2.882677
3.142148 3.424974]';
expdata=[0 10425.37 12723.77 16399.96 17158.68
20517.69 25343.61 30538.38 34726.26 37314.98 39768.33
43790.19 47506.67 49910.75 55692.83 56281.4 46442.42
52572.11 52047.22 57894.11 57611.62 64044.35 65610.49
68344.34 73008.01 78265 83632.1 88987.9 94336.64 99972.76
106802.6 114539 123166.9 131871 141309.6 153551.4
165299.4 175279.9 187812.7 200590.6 220207.4 251397
289119.8]';

```

```

end

% differential equation solver to get model prediction of viscosity
vs time

[t,etatemp,diverge]=extensional_pde_solver(r,wt,sigma,CI,para(1,:),para
(2,:),para(3,:),timedata,modes,re,orient_ext);

if diverge==1 %keep track of divergence
    tdiv=1;
    terr(m)=inf;
elseif diverge==0 %if no divergence, calculate error between model
and experiment
    for k=2:length(timedata)
        terr(m)=terr(m)+(log10(expdata(k))-log10(etatemp(k)))^2;
    end
end

%plot the model predictions and experimental data
if m==1
    figure
    if diverge==0
        loglog(t,etatemp,'r',timedata,expdata,'*r');
    end
    hold on
elseif m==2
    if diverge==0
        loglog(t,etatemp,'g',timedata,expdata,'*g');
    end
elseif m==3
    if diverge==0
        loglog(t,etatemp,'b',timedata,expdata,'*b');
    end
elseif m==4
    if diverge==0
        loglog(t,etatemp,'c',timedata,expdata,'*c');
    end
elseif m==5
    if diverge==0
        loglog(t,etatemp,'k',timedata,expdata,'*k');
    end
end

title('Extensional Viscosity HHT 2wt%');
xlabel('Time (s)');
ylabel('Viscosity (Pa*s)');
legend('Mod \epsilon=.01','Exp \epsilon=.01','Mod \epsilon=.03','Exp \epsilon=.03','Mod \epsilon=.1','Exp \epsilon=.1','Mod \epsilon=.3','Exp \epsilon=.3','Mod \epsilon=1','Exp \epsilon=1',-1);
set(gcf,'Units','normalized',
'WindowStyle','docked','OuterPosition',[0 0 1 1]);
hold off
end
end

Terror=sum(terr); %sum up error from all extension rates

```

```

%%
%%Part 2: Shear plotting
serr=zeros(1,5); %error per shear rate in model predictions
sdiv=0; %keeps track of any divergences in shear model predictions

for w=1:5 % w designates the w-th shear rate
    if w==1
        r=.01;
        %set .01(1)
        timedata=[0 0.02 0.03 0.04 0.05 0.06 0.07
0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15 0.16
0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.24 0.25
0.26 0.27 0.28 0.29 0.3 0.31 0.32 0.33 0.34
0.35 0.36 0.37 0.38 0.39 0.4 0.41 0.42 0.43
0.44 0.45 0.46 0.47 0.48 0.49 0.5 0.51 0.52
0.53 0.54 0.55 0.56 0.57 0.58 0.59 0.6 0.61
0.62 0.65 0.7 0.75 0.8 0.855 0.915 0.975 1.04 1.115
1.195 1.28 1.37 1.465 1.565 1.675 1.795 1.92 2.05
2.19 2.345 2.51 2.685 2.875 3.08 3.295 3.525 3.77
4.03 4.315 4.62 4.94 5.285 5.655 6.05 6.475 6.93
7.42 7.94 8.495 9.09 9.725 10.41 11.14 11.92 12.755
13.65 14.61 15.63 16.725 17.9 19.155 20.495 21.93 23.47
25.115 26.875 28.76 30.775 32.935 35.245 37.715 40.36 43.19
46.22 49.46 52.925 56.635 60.61 64.86 69.405 74.27 79.48
85.055 91.02 97.4 104.22 111.53 119.36 127.73 136.68 146.27
156.52 167.5 179.24 191.82 205.26 219.65 235.06 251.54 269.18
288.06 318.7 341.04 364.96 390.55 417.94 447.24 478.61 512.17
548.08 586.52 627.64 671.66 718.76 769.16 823.09 880.82 942.58
1008.7 1079.4 1155.1 1236.1 1322.8 1415.5 1514.8 1621 1734.7
1856.3 1986.5 2125.8 2274.9 2434.4 2605.1 2787.8 2983.3 3192.5
3416.3 3655.9 3912.3 4186.6 4480.2 4794.3 5130.5 5490.3 5875.3
6287.3 6728.2 7200];
        expdata=[0 442.64 1012.6 1768.6 2905 3693.3 4019.2
4837.7 5707.1 6051.3 6424 6934.2 7417.4 7639.1 8243.1 8756.5
8951.2 9213.4 9683.1 10026 10196 10692 11106 11223 11343
12047 12375 12229 12497 13078 13134 13283 13768 14100
13909 14075 14593 14733 14758 15145 15393 15522 15706
15976 16100 16236 16487 16822 16940 17219 17450 17269
17178 17706 18079 18081 18052 18307 18582 18720 18925
19368 19545 20145 20880 21524 22221 22951 23539 24236
24983 25724 26478 27178 27925 28723 29389 30215 30926
31721 32379 33212 33891 34669 35378 36193 36878 37659
38397 39094 39824 40522 41229 41905 42581 43234 43880
44549 45106 45687 46292 46859 47401 47898 48191 48606
49024 49518 50068 50641 51214 51859 52474 53114 53550
53781 54085 54345 54495 54668 55090 55401 55627 55698
55979 56407 56502 56358 56159 56863 57508 57383 57255
57510 57540 57767 57349 58001 58283 58119 58154 58100
58251 58316 58445 58237 58426 58546 58330 58434 58483
58326 58528 58243 58567 58359 58451 58369 57997 58063
58251 57987 57875 57986 57889 57863 57985 57750 57731
58025 57827 57892 58248 58222 58181 58610 58435 58570
58417 58520 58417 58313 58573 58589 58580 58899 58822
59262 59322 59461 59233 58792 58723 58762 58858 59021
58702 58253 58316 58514];
    end
end

```

```

elseif w==2
    r=.1;
    %set .1(3)
    timedata=[0 0.01 0.02 0.03 0.04 0.05 0.06
0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15
0.16 0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.24
0.25 0.26 0.275 0.305 0.35 0.4 0.455 0.515 0.585
0.665 0.755 0.86 0.975 1.105 1.255 1.425 1.62 1.845
2.1 2.39 2.715 3.085 3.51 3.99 4.54 5.165 5.87
6.675 7.595 8.635 9.82 11.17 12.7 14.445 16.43 18.685
21.255 24.175 27.495 31.275 35.57 40.455 46.01 52.33 59.52
67.695 76.995 87.575 99.605 113.28 128.85 146.55 177.4 201.77
229.49 261.01 296.88 337.66 384.05 436.82 496.82 565.08 642.71
731.01 831.43 945.66 1075.6 1223.4 1391.4 1582.6 1800];
    expdata=[0 47.56 606.43 1449.7 2348.5 3178.3 3978.3
4677.2 5311.3 5910.9 6447.8 6956.1 7433.9 7911 8342.7 8777.3
9160.9 9561.8 9965.6 10310 10633 10989 11310 11617 11936
12170 12487 12903 13640 14766 15862 16922 18042 19203
20436 21708 23068 24427 25852 27324 28686 30031 31401
32729 34119 35448 36699 38064 39314 40295 41652 42824
43781 44732 45417 46413 47050 47539 48099 48349 48675
48741 48758 48700 48503 48247 47970 47606 47242 46890
46533 46219 45951 45749 45587 45514 45503 45427 45343
45249 45145 45181 45302 45257 45066 44909 44915 44746
44712 44278 44189 43827 43584 43794 43760 43936];
elseif w==3
    r=.3;
    %set .3(3)
    timedata=[0 0.01 0.02 0.03 0.04 0.05 0.06
0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15
0.16 0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.24
0.25 0.26 0.275 0.305 0.345 0.39 0.445 0.505 0.57
0.65 0.74 0.84 0.955 1.085 1.23 1.395 1.59 1.81
2.055 2.335 2.65 3.01 3.425 3.89 4.42 5.025 5.71
6.49 7.38 8.39 9.535 10.835 12.315 14 15.915 18.09
20.56 23.375 26.575 30.205 34.335 39.03 44.365 50.43 57.325
65.165 74.075 84.205 95.72 108.81 123.69 140.6 159.83 193.32
219.75 249.8 283.96 322.79 366.93];
    expdata=[0 66.031 624.41 1415 2256.7 3048.4 3757.8
4418.8 5029.6 5591.8 6115.1 6603.6 7075.5 7510.8 7905.8 8311
8684.8 9049.8 9405.8 9731.3 10069 10373 10663 10960 11241
11512 11779 12170 12897 13781 14677 15656 16606 17564
18662 19842 21006 22105 23166 24286 25339 26509 27518
28569 29665 30583 31487 32281 33070 33657 34197 34532
34771 34785 34647 34332 33770 33081 32301 31465 30681
30001 29422 28873 28377 27963 27606 27475 27409 27087
26845 26816 26650 26670 26531 26342 25856 25536 25542
25261 25432 25555 25623 25484];
elseif w==4
    r=1;
    %set 1(1)
    timedata=[0 0.01 0.02 0.03 0.04 0.05 0.06
0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15
0.16 0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.24
0.25 0.26 0.27 0.29 0.325 0.37 0.42 0.475 0.54
0.615 0.7 0.795 0.9 1.02 1.155 1.31 1.485 1.685 1.915
2.17 2.46 2.79 3.165 3.59 4.075 4.625 5.245 5.95

```

```

6.755    7.665    8.695    9.87    11.2    12.71    14.425    16.365    18.57
21.075    23.915    27.135    30.79    34.94    39.65    44.995    51.06    57.945
65.755];
    expdata=[0    87.354    628.09    1384.5    2182.5    2935    3619.7
4239.6    4826.3    5366.2    5836.6    6320.1    6761.8    7166.7    7542.3    7928.6
8241.4    8592.8    8879.2    9196.6    9512.9    9765.7    10042    10323    10559
10822    11075    11311    11747    12442    13335    14168    14970    15923
16845    17730    18594    19405    20269    21011    21711    22367    22907
23391    23716    23927    24009    23952    23764    23433    23011    22523
22006    21492    20996    20534    20078    19625    19248    18957    18707
18433    18248    18210    18207    18220    18286    18058    17770    17614
17452    17137];
elseif w==5
    r=3;
    %set 3(3)
    timedata=[0    0.01    0.02    0.03    0.04    0.05    0.06
0.07    0.08    0.09    0.1    0.11    0.12    0.13    0.14    0.15
0.16    0.17    0.18    0.19    0.2    0.21    0.22    0.23    0.24
0.25    0.26    0.27    0.295    0.335    0.38    0.435    0.495    0.56
0.635    0.72    0.815    0.925    1.05    1.19    1.35    1.535    1.745
1.98    2.245    2.545    2.89    3.285    3.73    4.235    4.81    5.46
6.2    7.045    8    9.08    10.315    11.715    13.3    15.105    17.155    19.485
22.13    25.135    28.545    32.415    36.815];
    expdata=[0    82.036    606.86    1335.9    2110.6    2844.7    3506.9
4095.5    4625.8    5119.7    5589.3    6035.9    6411.6    6741.9    7117.2    7440.7
7745.2    8032.4    8299.3    8550.5    8808.2    9026.2    9261.2    9544    9730.8
9949.6    10173    10344    10786    11417    12030    12639    13152    13567
13980    14230    14400    14420    14353    14139    13851    13457    13004
12563    12119    11723    11333    11006    10731    10514    10373    10274
10158    10107    10166    10208    10270    10335    10306    10145    10120
10129    10067    10013    9953.8    9964.8    10036];
end

%shift time data back to account for start up delay in
rheometer
factor=timedata(2)-0.0005;
timedata=timedata-factor;
timedata(1)=0;

%differential equation solver to get model prediction of
viscosity vs time

[t,etatep,diverge]=shear_pde_solver(r,wt,sigma,CI,para(1,:),para(2,:),
para(3,:),timedata,modes,re,orient_shr);

if diverge==1 %keep track of divergence
    sdiv=1;
    serr(w)=inf;
elseif diverge==0 %if no divergence, calculate error between
model and experiment
    for k=2:length(timedata)
        serr(w)=serr(w)+(log10(expdata(k))-
log10(etatep(k)))^2;
    end
end

%plot the model predictions and experimental data
if w==1

```

```

figure
if diverge==0
    loglog(timedata,expdata,'*',t,etatemp,'Color',[0,0,1]);
end
hold on
elseif w==2
    if diverge==0
        loglog(timedata,expdata,'*',t,etatemp,'Color',[0,0,0]);
    end
elseif w==3
    if diverge==0
        loglog(timedata,expdata,'*',t,etatemp,'Color',[0,1,1]);
    end
elseif w==4
    if diverge==0
        loglog(timedata,expdata,'*',t,etatemp,'Color',[.6,0,.9]);
    end
elseif w==5
    if diverge==0
        loglog(timedata,expdata,'*',t,etatemp,'Color',[.7,.9,.3]);
    end

    xlabel('Time (s)')
    ylabel('Viscosity (Pa*s)')
    title('Shear Viscosity HHT 2wt%')
    legend('Exp \gamma=0.01','Mod \gamma=0.01','Exp \gamma=0.1','Mod \gamma=0.1','Exp \gamma=.3','Mod \gamma=.3','Exp \gamma=1','Mod \gamma=1','Exp \gamma=3','Mod \gamma=3',-1);
    set(gcf,'Units','normalized',
'WindowStyle','docked','OuterPosition',[0 0 1 1]); %this places the
plot in a maximized window
    hold off
end
end
Error=sum(serr); %sum up error from all shear rates

%%
%%PART 3: SAOS plotting
%read in sigma and CI values to be sent to other programs
%call SAOSModeling variant to calc experimental stress.
%call SAOSFittingGpGdp variant to calc model prediction.
%find error between the two and send back as Gerror

%NOTE: diverging probably kills the programs this subfunction calls

Gerror=0; %error in SAOS model predictions
gdiv=0; %keeps track of any divergences in SAOS model predictions

%data from Koki Data_Solvent Casting
freq=[ 100      63.096  39.811  25.119  15.849  10      6.3096  3.9811
2.5119  1.5849  1      0.63096 0.39811 0.25119 0.15849 0.1
0.063096 0.039811 0.025119 0.015849 0.01];
%storage modulus

```

```

GpExp=[109980    95781    82077    69679    58066    47737    38316    30197
23154    17170    12300    8915.7    5835.3    4025.7    2576.3    1613.8    967.34
549.9         310.61    167.18         102.05];
%loss modulus
GdpExp=[55997    50642    45518    41323    37171    33418    29749    25877
22421    18612    15694    12371    9995.4    7626.9    5584.5    4093.9    2849.3
2075.2         1420.2    903.06         619.69];

% this commented-out section calls on 'SAOSModeling' which calculates
and
% plots the stress wave from SAOS flow. Uncomment if you desire to see
% this, but only use every third point from the experimental data;
% otherwise, the computations are way too many and tedious and MATLAB
struggles
%           [export] =
SAOSModeling(para,sigma,CI,freq,GpExp,GdpExp,modes,graph);
%           time=export(:,1);
%           taucl2=export(:,2);
%           pred=export(:,3);

graph=0; %0 for no plotting in SAOSFittingGpGdp, =1 for
plotting
%fit G' G'' of composite to SAOS stress wave
[GpGdp,diverge] =
SAOSFittingGpGdp(para,sigma,CI,freq,GpExp,GdpExp,wt,modes,graph,re,orie
nt_shr);
GpMod=GpGdp(:,1);
GdpMod=GpGdp(:,2);

if diverge==0 %if no divergence, calculate error between model
and experiment
for j=1:length(freq)
    Gerror=Gerror+(log10(GpExp(j))-
log10(GpMod(j)))^2+(log10(GdpExp(j))-log10(GdpMod(j)))^2;
end
%plot the model predictions and experimental data
figure

loglog(freq,GpExp,'b*',freq,GdpExp,'g*',freq,GpMod,'b',freq,GdpMod,'g')
legend('GpExp','GdpExp','GpMod','GdpMod',-1)
title('SAOS HHT 2wt%')
else %keep track of divergence
    Gerror=inf;
    gdiv=1;
end

%%
% uncomment to check errors and divergences if interested
% Terror
% Serror
% Gerror
% tdiv
% sdiv
% gdiv

end

```

Composite Model Predictions: Solvent Cast with O-CNFs (SC)

```
function SC_2wt
%This program plots the viscosity for the SC 2wt% composite for
transient
%extension and transient shear as well as moduli predictions for SAOS
% given the optimized values for sigma and CI and hfactor

sigma=.5;
CI=.0499;
hfactor=1.76;

para=[ 1563.180023    10503.23358 30793.85272 9864.433234 9310.030642
40690.01669; %optimized values for etap by mode
      0.01461114    0.235979496 3.210427777 34.39722848 6390.005373
116255.5053; %optimized values for lambda by mode
      0.818868357 0.56          0.56          0.078601046 0.025189799
0.001841609]; %optimized values for alpha by mode
modes=6;
wt=.02; %mass fraction of cnfs
re=69/hfactor; %69= number ave length, 124= weight ave length
orient_ext=[.528 .236 .236]; %experimentally determined initial
orientation of fibers (a11,a22,a33)
orient_shr=[.528 0 .236]; %experimentally determined initial
orientation of fibers (a11,a12,a22)

%%
%%Part 1: Extensional plotting
terr=zeros(1,5); %error per extension rate in model predictions
tdiv=0; %keeps track of any divergences in extensional model
predictions

for m=1:5 % m designates the m-th extension rate
    % see data for poster for all sets which have been truncated
    if m==1
        r=.01;
        %set .01(1) slight trunc at beginning
        timedata=[0 22.25409    26.64752    31.90831    38.20769
45.7507 54.78287    65.59817    78.54865    94.05583    112.6245
134.8589 161.483 193.3631    231.5372    277.2475    331.9821
397.5225 476.0019    569.9748    682.5]';
        expdata=[0 387843.5    377964.3    376909.3    400561.4
416749.4 430228.8    446005.1    453425.2    464266.9    480019.8
481094.1 496619.5    525034.7    549660.1    602283.9    686062.9
759297.2 943786.7    1428359 2798893]';
    elseif m==2
        r=.03;
        %set .03(1) trunc
        timedata=[0 9.700294    11.35781    13.29856    15.57092
18.23158 21.34686    24.99447    29.26535    34.26601    40.12115
46.97677 55.00384    64.40251    75.40717    88.29222    103.379
121.0437 141.7268    165.9441    194.2995]';
        expdata=[0 318799.5    322584.2    331946.3    351364.5
369053 384363.7    395667.4    412336.7    428658.5    443033.7
```



```

447931.5    464958.8    467193.4    474136.4    481202.3    510103.9
600216.2    745073.8    1025986 1601637]';
elseif m==3
    r=.1;
    %set .1(2)
    timedata=[0 3.190139    3.644588    4.163775    4.756922
5.434566    6.208744    7.093206    8.103663    9.258065    10.57692
12.08364    13.80501    15.77159    18.01832    20.58511    23.51755
26.86772    30.69514    35.0678 40.06336    45.77055    52.29077
59.73981]';
    expdata=[0 193777.5    206943.7    212962.1    220398.4
230104.9    238061.3    247736.3    258577.8    262897.8    268821.3
288701.6    313619.3    334114.8    360457.6    375016.1    392091.5
395367.7    401158 363448.8    307047.8    284038.5    260705.1
551346.6]';
elseif m==4
    r=.3;
    %set .3(4)
    timedata=[0 0.1 0.1117125    0.1247968    0.1394135
0.1557423    0.1739836    0.1943614    0.2171259    0.2425567    0.2709661
0.3027029    0.3381569    0.3777634    0.4220089    0.4714366    0.5266534
0.5883376    0.6572464    0.7342262    0.8202223    0.9162906    1.023611
1.143501    1.277433    1.427052    1.594195    1.780915    1.989504
2.222524    2.482837    2.773638    3.0985 3.461411    3.866828
4.319729    4.825676    5.390882    6.022287    6.727646    7.51562
8.395884    9.37925 10.47779    11.705 13.07595]';
    expdata=[0 15703.5 19405.89    23630.79    30351.06
39551.26    47271.38    48617.73    55422.28    63423.53    67141.63
76307.47    79001.23    87821.25    95223.06    102599.8    109690.8
115282.6    121508.2    126849.7    130611.6    134323.6    138513.3
143107.4    149151.3    155482.3    161784.5    165403.1    170278.8
176591.8    183432.5    188674.6    190785.9    197256.5    203240.9
215745.1    221730.9    226618.8    240972.5    242744.7    265836.2
274835.6    300677.2    319750 377306.3    380382.5]';
elseif m==5
    r=1;
    %set 1(2)
    timedata=[0 0.1 0.109001    0.1188123    0.1295066
0.1411636    0.1538698    0.1677197    0.1828162    0.1992716    0.2172081
0.2367591    0.2580699    0.2812989    0.3066187    0.3342176    0.3643007
0.3970916    0.432834    0.4717935    0.5142599    0.5605487    0.6110039
0.6660007    0.7259477    0.7912906    0.862515    0.9401504    1.024774
1.117014    1.217557    1.32715 1.446607    1.576817    1.718747
1.873452    2.042083    2.225892    2.426245    2.644632    2.882677
3.142148    3.424974    3.733258    4.06929 4.435569    4.834816
5.27]';
    expdata=[0 13700.15    19522.7 21038.07    23982.51
28076.44    33713.48    39973.21    44765.05    47850.18    51608.97
56171.65    60023.85    63253.41    66884.19    69822.2 73230.11
76699.03    79586.54    83821.8 87150.84    90598.12    95051.02
99202.08    103614.5    108540.6    113448 118565.5    124469.2
128392.1    132122.8    141084.5    146724.3    150072.6    157853.2
171811.9    181501 193433.8    209473.4    221650.6    238442
257288.8    284907.1    311394.4    335601.6    344309.1    364330
383190.8]';
end

```

```

    %differential equation solver to get model prediction of viscosity
    vs time

[t,etatemp,diverge]=extensional_pde_solver(r,wt,sigma,CI,para(1,:),para
(2,:),para(3,:),timedata,modes,re,orient_ext);

    if diverge==1 %keep track of divergence
        tdiv=1;
        terr(m)=inf;
    elseif diverge==0 %if no divergence, calculate error between model
and experiment
        for k=2:length(timedata)
            terr(m)=terr(m)+(log10(expdata(k))-log10(etatemp(k)))^2;
        end
    end

%plot the model predictions and experimental data
if m==1
    figure
    if diverge==0
        loglog(t,etatemp,'r',timedata,expdata,'*r');
    end
    hold on
elseif m==2
    if diverge==0
        loglog(t,etatemp,'g',timedata,expdata,'*g');
    end
elseif m==3
    if diverge==0
        loglog(t,etatemp,'b',timedata,expdata,'*b');
    end
elseif m==4
    if diverge==0
        loglog(t,etatemp,'c',timedata,expdata,'*c');
    end
elseif m==5
    if diverge==0
        loglog(t,etatemp,'k',timedata,expdata,'*k');
    end

    title('Extensional Viscosity SC 2wt%');
    xlabel('Time (s)');
    ylabel('Viscosity (Pa*s)');
    legend('Mod \epsilon=.01','Exp \epsilon=.01','Mod
\epsilon=.03','Exp \epsilon=.03','Mod \epsilon=.1','Exp
\epsilon=.1','Mod \epsilon=.3','Exp \epsilon=.3','Mod \epsilon=1','Exp
\epsilon=1',-1);
    set(gcf,'Units','normalized',
'WindowStyle','docked','OuterPosition',[0 0 1 1]); %this places the
plot in a maximized window
    hold off
end
end
Error=sum(terr); %sum up error from all extension rates

```

```

%%
%%Part 2: Shear plotting
serr=zeros(1,7); %error per shear rate in model predictions
sdiv=0; %keeps track of any divergences in shear model predictions

for w=1:5 % w designates the w-th shear rate
    if w==1
        r=.01;
        %set .01(1)
        timedata=[0 0.01 0.02 0.03 0.04 0.05 0.06
0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15
0.16 0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.24
0.25 0.26 0.28 0.315 0.36 0.41 0.465 0.53 0.6
0.68 0.775 0.885 1.01 1.145 1.3 1.48 1.685 1.92
2.185 2.485 2.83 3.22 3.665 4.175 4.75 5.405 6.155
7.005 7.975 9.08 10.335 11.765 13.39 15.245 17.36 19.76
22.495 25.61 29.155 33.19 37.785 43.02 48.975 55.755 63.475
72.26 82.265 93.655 106.62 121.38 138.19 157.33 190.71 217.11
247.17 281.4 320.36 364.73 415.22 472.72 538.18 612.7 697.53
794.13 904.08 1029.3 1171.8 1334 1518.8 1729.1 1968.5 2241
2551.4 2904.6 3306.8 3764.7 4286 4879.5 5555.1 6324.3];
        expdata=[0 60.133 453.29 966.59 1866.4 2873 3489.5
4050.3 4782.7 5327.9 5575.5 6204.6 6852.1 7106.4 7438.6 8003.8
8383 8588.7 9011.8 9586.3 9829.8 10029 10495 10881 10928
11278 11680 12102 13014 14059 15124 16204 17375 18522
19719 20984 22344 23705 25013 26405 27851 29307 30754
32264 33690 35183 36643 38133 39581 40968 42309 43617
44910 46120 47255 48228 49075 50146 51379 52774 54190
54938 55641 56217 57085 57696 58585 58978 59304 60680
60866 61582 61777 62925 63440 63962 64578 65040 65442
65784 66052 65968 65853 65292 64631 64033 63405 62448
61320 60616 59885 59125 58773 58508 58161 57946 57719
57776 58681 59166 60205 60529 60855 60054 60654];
    elseif w==2
        r=.1;
        %set .1(2)
        timedata=[0 0.01 0.02 0.03 0.04 0.05 0.06
0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15
0.16 0.17 0.18 0.19 0.2 0.21 0.22 0.23 0.24
0.25 0.26 0.275 0.305 0.35 0.4 0.455 0.515 0.585
0.665 0.755 0.86 0.975 1.105 1.255 1.425 1.62 1.845
2.1 2.39 2.715 3.085 3.51 3.99 4.54 5.165 5.87
6.675 7.595 8.635 9.82 11.17 12.7 14.445 16.43 18.685
21.255 24.175 27.495 31.275 35.57 40.455 46.01 52.33 59.52
67.695 76.995 87.575 99.605 113.28 128.85 146.55 177.4 201.77
229.49 261.01 296.88 337.66 384.05 436.82 496.82 565.08 642.71
731.01 831.43 945.66 1075.6 1223.4 1391.4 1582.6 1800];
        expdata=[0 46.69 534.64 1332.6 2176.3 2964.8 3727.8
4412.9 5013.3 5591.9 6170.2 6691.2 7173.6 7642.8 8091.4 8493.5
8883.1 9289.5 9642.4 9972 10353 10704 11007 11334 11646
11934 12221 12630 13447 14563 15682 16835 17961 19139
20369 21656 22966 24300 25739 27163 28570 30014 31453
32987 34555 35976 37305 38609 40282 41632 42792 43995
45096 45865 46987 47621 48093 48491 48618 48453 48145
47535 46686 45747 44626 43371 42111 40947 40067 39452
38987 38733 38722 38947 39274 39675 39927 40047 40079

```

```

40401    40765    40633    40822    40897    41109    41582    41733    41999
42335    42222    42799    42591    42464    42640    43358    44800];
elseif w==3
    r=.3;
    %set .3(1)
    timedata=[0 0.01    0.02    0.03    0.04    0.05    0.06
0.07    0.08    0.09    0.1 0.11    0.12    0.13    0.14    0.15
0.16    0.17    0.18    0.19    0.2 0.21    0.22    0.23    0.24
0.25    0.26    0.275    0.305    0.345    0.39    0.445    0.505    0.57
0.65    0.74    0.84    0.955    1.085    1.23    1.395    1.59    1.81
2.055    2.335    2.65    3.01    3.425    3.89    4.42    5.025    5.71
6.49    7.38    8.39    9.535    10.835    12.315    14    15.915    18.09
20.56    23.375    26.575    30.205    34.335    39.03    44.365    50.43    57.325
65.165    74.075    84.205    95.72    108.81    123.69    140.6    159.83    193.32
219.75    249.8    283.96    322.79    366.93    417.11    474.15    538.99    612.7
696.48    791.73    900.01];
    expdata=[0 1.7318    546.57    1358.4    2224.9    3051.1    3824.1
4527.7    5180    5776.1    6333.1    6842.5    7329.9    7808.9    8244.6    8681.6
9047.8    9481.8    9860.3    10224    10572    10907    11203    11517    11832
12153    12453    12874    13704    14730    15835    17078    18362    19554
20856    22145    23421    24779    26081    27376    28694    29895    31154
32385    33357    34290    35010    35745    36162    36356    36445    36218
35860    35244    34470    33662    32841    32027    31258    30592    30145
29850    29780    29898    30123    30304    30329    30249    30213    30149
29944    29640    29198    28897    28697    28995    29499    30168    30387
29750    29243    29098    30009    30418    30073    30110    30002    30242
29961    30404    31371];
elseif w==4
    r=1;
    %set 1(1)
    timedata=[0 0.01    0.02    0.03    0.04    0.05    0.06
0.07    0.08    0.09    0.1 0.11    0.12    0.13    0.14    0.15
0.16    0.17    0.18    0.19    0.2 0.21    0.22    0.23    0.24
0.25    0.26    0.27    0.29    0.325    0.37    0.42    0.475    0.54
0.615    0.7 0.795    0.9 1.02    1.155    1.31    1.485    1.685    1.915
2.17    2.46    2.79    3.165    3.59    4.075    4.625    5.245    5.95
6.755    7.665    8.695    9.87    11.2    12.71    14.425    16.365    18.57
21.075    23.915    27.135    30.79    34.94    39.65    44.995    51.06    57.945
65.755    74.62    84.68    96.09    109.04    123.74    140.43    159.35];
    expdata=[0 83.562    635.88    1426    2283    3094.4    3843.7
4523.4    5155.8    5733.9    6312.9    6823.6    7291.8    7762    8188    8612.7
9032.3    9384.7    9755.5    10029    10417    10735    11029    11349    11635
11933    12268    12515    13071    13947    14939    15939    16941    17989
18969    19988    20965    21849    22637    23360    23937    24322    24533
24502    24247    23711    22939    21997    20968    19850    18720    17770
17262    17162    17303    17616    18041    18401    18610    18591    18525
18315    18014    17718    17554    17912    18391    18547    18259    18296
18344    18484    18327    18469    18667    18545    18048    17298    17227];
elseif w==5
    r=3;
    %set 3(2)
    timedata=[0 0.01    0.02    0.03    0.04    0.05    0.06
0.07    0.08    0.09    0.1 0.11    0.12    0.13    0.14    0.15
0.16    0.17    0.18    0.19    0.2 0.21    0.22    0.23    0.24
0.25    0.26    0.27    0.295    0.335    0.38    0.435    0.495    0.56
0.635    0.72    0.815    0.925    1.05    1.19    1.35    1.535    1.745
1.98    2.245    2.545    2.89    3.285    3.73    4.235    4.81    5.46

```

```

6.2 7.045 8 9.08 10.315 11.715 13.3 15.105 17.155 19.485
22.13 25.135 28.545 32.415 36.815 41.815];
expdata=[0 71.359 585.85 1288.8 2060.1 2807 3484
4112.1 4690.9 5193.1 5668.5 6120.5 6540.9 6912.2 7291.7 7589.7
7944.7 8231.9 8528.3 8813.8 9118.9 9353.9 9592 9839.8 10067
10278 10477 10665 11086 11760 12342 12867 13415 13797
14099 14302 14417 14446 14352 14110 13858 13506 13146
12793 12486 12169 11887 11663 11463 11275 11086 10901
10753 10580 10532 10572 10461 10248 10124 10108 10275
9986.5 9727.1 9464.9 9189.8 9304.7 9346.9 9218.6];
end
%shift time data back to account for start up delay in
rheometer
factor=timedata(2)-0.0005;
timedata=timedata-factor;
timedata(1)=0;
%differential equation solver to get model prediction of
viscosity vs time

[t,etatep,diverge]=shear_pde_solver(r,wt,sigma,CI,para(1,:),para(2,:),
para(3,:),timedata,modes,re,orient_shr);

if diverge==1 %keep track of divergence
sdiv=1;
serr(w)=inf;
elseif diverge==0 %if no divergence, calculate error between
model and experiment
for k=2:length(timedata)
serr(w)=serr(w)+(log10(expdata(k))-
log10(etatep(k)))^2;
end
end

%plot the model predictions and experimental data
if w==1
figure
if diverge==0
loglog(timedata,expdata,'*',t,etatep,'Color',[0,0,1]);
end
hold on
elseif w==2
if diverge==0
loglog(timedata,expdata,'*',t,etatep,'Color',[0,0,0]);
end
elseif w==3
if diverge==0
loglog(timedata,expdata,'*',t,etatep,'Color',[0,1,1]);
end
elseif w==4
if diverge==0
loglog(timedata,expdata,'*',t,etatep,'Color',[.6,0,.9]);
end
elseif w==5
if diverge==0
loglog(timedata,expdata,'*',t,etatep,'Color',[.7,.9,.3]);
end
end

```

```

        xlabel('Time (s)')
        ylabel('Viscosity (Pa*s)')
        title('Shear Viscosity SC 2wt%')
        legend('Exp \gamma=0.01','Mod \gamma=0.01','Exp
\gamma=0.1','Mod \gamma=0.1','Exp \gamma=.3','Mod \gamma=.3','Exp
\gamma=1','Mod \gamma=1','Exp \gamma=3','Mod \gamma=3',-1);
        set(gcf,'Units','normalized',
'WindowStyle','docked','OuterPosition',[0 0 1 1]);
        hold off
    end
end
Error=sum(serr); %sum up error from all shear rates

%%
%%PART 3: SAOS
%read in sigma and CI values to be sent to other programs
%call SAOSModeling variant to calc experimental stress.
%call SAOSFittingGpGdp variant to calc model prediction.
%find error between the two and send back as Gerror

%NOTE: diverging probably kills the programs this subfunction calls

Gerror=0; %error in SAOS model predictions
div=0; %keeps track of any divergences in SAOS model predictions

%data from Koki Data Solvent Casting
freq=[100    63.096    39.811    25.119    15.849    10    6.3096    3.9811    2.5119
1.5849    1    0.63096    0.39811    0.25119    0.15849    0.1    0.063096    0.039811
0.025119    0.015849    0.01 ];
%storage modulus
GpExp=[137280    119870    103510    88257    74226    60987    49437    38805
30013    22372    16364    11517    8185.3    5409.5    3205.8    1938.1    1320.8
689.58    421.55    236.35    109.48];
%loss modulus
GdpExp=[62302    57776    53664    49458    45444    40648    36617    32468
28097    23851    19700    16027    12551    9515.1    7175.4    5241.3    3768.1
2639.6    1749.3    1157.1    757.66];

% this commented-out section calls on 'SAOSModeling' which calculates
and
% plots the stress wave from SAOS flow. Uncomment if you desire to see
% this, but only use every third point from the experimental data;
% otherwise, the computations are way too many and tedious and MATLAB
struggles
%
[export] =
SAOSModeling(para,sigma,CI,freq,GpExp,GdpExp,modes,graph);
%
time=export(:,1);
%
tauc12=export(:,2);
%
pred=export(:,3);

graph=0; %=0 for no plotting in SAOSFittingGpGdp, =1 for
plotting
%fit G' G'' of composite to SAOS stress wave

```

```

        [GpGdp,diverge] =
SAOSFittingGpGdp(para,sigma,CI,freq,GpExp,GdpExp,wt,modes,graph,re,orient_shr);
        GpMod=GpGdp(:,1);
        GdpMod=GpGdp(:,2);

        if diverge==0 %if no divergence, calculate error between model
and experiment
            for j=1:length(freq)
                Gerror=Gerror+(log10(GpExp(j))-
log10(GpMod(j)))^2+(log10(GdpExp(j))-log10(GdpMod(j)))^2;
            end
            %plot the model predictions and experimental data
            figure

loglog(freq,GpExp,'b*',freq,GdpExp,'g*',freq,GpMod,'b',freq,GdpMod,'g')
            legend('GpExp','GdpExp','GpMod','GdpMod',-1)
            title('SAOS SC 2wt%')
        else %keep track of divergence
            Gerror=inf;
            gdiv=1;
        end

%%
% uncomment to check errors and divergences if interested
% Terror
% Serror
% Gerror
% tdiv
% sdiv
% gdiv
end

```

Composite Model Predictions: Fitting G' and G''

```
function [GpGdp,div] =
SAOSFittingGpGdp(para,sigma,CI,freq,Gpo,Gdpo,wt,modes,graph,re,orient)
%This program models the stress wave from a small amplitude oscillatory
%shear flow and fits the optimum values of G' and G'' to this stress
wave
% calls on SAOSimulGsfit to calc composite stress wave

%From Tim Kremer: see his thesis for details on this program

format long

%initial guess of G' and G''
guess=[0 35.06125 118.77715 218.069 403.069 731.4956667 1295.72
2281.803333 3723.183333 5670.693333 8656.983333 12385.93333 17431.8
23216.63333 30205.2 38639.13333 47962.26667 58520.46667 70267.56667
82679.76667 96278.46667;
0 532.1383333 799.4886667 1259.376667 1859.03 2733.77 3907.66
5450.736667 7393.43 9767.963333 12548.53333 15681.63333 19139.5 22729.5
26576.8 30379.83333 34434.86667 38387.86667 42793.26667 47297.4
52348.03333];

LB=zeros(2,1); %lower boundary

UB=[]; %upper boundary

div=zeros(1,length(freq)); %divergence variable

for x=1:length(freq)

    x;
    %calculate model's prediction
    [stresst tspant
diverge]=SAOSSimulGsfit(freq(x),wt,sigma,CI,para,modes,re,orient);

    % if diverge==1
    %     div(x)=1;
    %     continue
    % end

    temp=find(tspant>=4*pi/freq(x),1);

    stress{x}=stresst(temp:end);
    tspan{x}=tspant(temp:end);

%options=optimset('Algorithm','interior-point');

[parameters fval exitflag] = fmincon(@Fitting,[guess(1,x)
guess(2,x)],[],[],[],[],LB,UB,[]);%,options);

Gpf(x)=parameters(1);
Gdpf(x)=parameters(2);
```



```

end

Gp;
Gdp;

%model predictions for stress wave
predfinal=Gpf(x)*0.5*sin(freq(x)*tspan{x})+Gdpf(x)*0.5*cos(freq(x)*tspan{x});
predo=Gpo(x)*0.5*sin(freq(x)*tspan{x})+Gdpo(x)*0.5*cos(freq(x)*tspan{x});

%plotting
if graph ==1
figure
plot(tspan{x},predfinal,tspan{x},stress{x},tspan{x},predo)
legend('Gprime Gdoubleprime','stress','original Gs')
size(predfinal)
size(stress{x})

%calculate r^2 value (regression coefficient) with stress as value
being compared to
C = corrcoef(stress{x},predfinal);
rsq1 = C(1,2).^2;

title(['Stress Wave, r^2 = ',num2str(rsq1)])
end

GpGdp=[Gpf' Gdpf'];

finaler=sum(er);

function error = Fitting(para)
%this subfunction calculates the error between model prediction and
%experiment

Gp=para(1);
Gdp=para(2);

error=0;

pred=Gp*0.5*sin(freq(x)*tspan{x})+Gdp*0.5*cos(freq(x)*tspan{x});
for i=1:length(tspan{x})
    error= error + (stress{x}(i)-pred(i))^2; %calculate error
end
er(x)=error;

end

end

```

Composite Model Predictions: Solving the Constitutive Model for SAOS

Flow

```
function [tauc12,tspan,diverge] =
SAOSSimulGsfit(w,mass,sigma,CI,para,modes,re,orientation)
%This program solves the constitutive model equations for oscillatory
shear
%flow and outputs composite stress wave in 12 direction. This is a
%specialized version of 'shear_pde_solver'. See'shear_pde_solver' for
more
%specific details

%From Tim Kremer: see his thesis for details on this program

p=0;
x=1;

r0=0.5; %percent

if mass==0
    sigma=1;
end

% re=33.0; %aspect ratio

chi=1.0*(re^2-1)/(re^2+1); %another form of aspect ratio
rf=1750.0; %fiber density
rs=1000.0; %polymer density

etap=para(1,:);
lambda=para(2,:);
alpha=para(3,:);

% modes=5;

tspan=0:1/6/w:8*pi/w;

phi=1.0*rs*mass/(rf+(rs-rf)*mass); %volume fraction
Ap=1.0*re^2/(3*log(sqrt(1.0*pi/phi))); %A2, shape factor

tau_final=zeros(3,modes);
diverge=0;

for x=1:modes
    initial=[0 0 0 0 orientation];
    [time, yo]=ode23tb(@modepolymersub,tspan,initial);

    asdf=length(yo(:,1));
    L=length(time);

    if x==1
        tau1=zeros(L,modes);
```

```

        tau12=zeros(L,modes);
        tau2=zeros(L,modes);
        tau3=zeros(L,modes);
    end

    if p==0
        tau1(:,x)=yo(:,1);
        fdsa=length(tau1);
        tau12(:,x)=yo(:,2);
        tau2(:,x)=yo(:,3);
        tau3(:,x)=yo(:,4);

        r=r0*w*cos(w*time);
%         tau_final(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x) tau3(L,x)]';

        p=p+1;
    elseif p>0
%         if asdf<fdsa
%             diverge=1;
%             break
%         end
        tau1(:,x)=yo(:,1);
        fdsa=length(tau1);
        tau12(:,x)=yo(:,2);
        tau2(:,x)=yo(:,3);
        tau3(:,x)=yo(:,4);

        r=r0*w*cos(w*time);
%         tau_final(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x) tau3(L,x)]';
    end
end

total11=sum(tau12,2);
total12=sum(tau12,2);
total22=sum(tau12,2);
total33=sum(tau12,2);

a11=yo(:,5);
a12=yo(:,6);
a22=yo(:,7);
a33=1-a11-a22;

eta=total12./r;
coef=2.0*eta*phi;

if r~=0
    %disp('r~=0')
    tf12=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-
a11-a22))).*(-1.0/35.0+1.0/7.0.*(a11+a22)))+(a12.^2).*(1-
27.*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
    tf11=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-
a11-a22))).*(3.0/7.0.*(a12)))+(a11.*a12).*(1-27*(a11.*a22.*(1-a11-
a22))+27.0.*(a12.^2).*(1-a11-a22)));

```

```

    tf22=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-
a11-a22))).*(3.0/7.0.*(a12)))+(a22.*a12).*(1-27*(a11.*a22.*(1-a11-
a22))+27.0.*(a12.^2).*(1-a11-a22)));
    tf33=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-
a11-a22))).*(1.0/7.0.*(a12)))+(1-a11-a22).*(a12).*(1-27*(a11.*a22.*(1-
a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
else
    %disp('r==0')
    tf12=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-
(a12.^2).*(1-a11-a22))).*(-1.0/35.0+1.0/7.0*(a11+a22)))+(a12.^2).*(1-
27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
    tf11=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-
(a12.^2).*(1-a11-a22))).*(3.0/7.0*(a12)))+(a11.*a12).*(1-
27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
    tf22=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-
(a12.^2).*(1-a11-a22))).*(3.0/7.0*(a12)))+(a22.*a12).*(1-
27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
    tf33=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-
(a12.^2).*(1-a11-a22))).*(1.0/7.0*(a12)))+(1-a11-a22).*(a12).*(1-
27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
end

tauc12=tf12+total12;
tauc11=tf11+total11;
tauc22=tf22+total22;
tauc33=tf33+total33;
etac=tauc12./r;
N1c=tauc11-tauc22;
N2c=tauc22-tauc33;
aijkl12=((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(-
1.0/35.0+1.0/7.0*(a11+a22)))+(a12.^2).*(1-27*(a11.*a22.*(1-a11-
a22))+27.0.*(a12.^2).*(1-a11-a22)));

function dyo = modepolymersub(t,y)

    tp11=y(1);
    tp12=y(2);
    tp22=y(3);
    tp33=y(4);
    a11=y(5);
    a12=y(6);
    a22=y(7);
    a33=1-a11-a22;
    dyo=zeros(7,1);

    dyo(1,1)=-alpha(x)*(tp11^2+tp12^2)/etap(x)-
sigma*tp11/lambda(x)+2*r0*w*cos(w*t)*tp12-3*(1-
sigma)*(tp11*a11+tp12*a12)/lambda(x);
    dyo(2,1)=-alpha(x)*(tp11*tp12+tp12*tp22)/etap(x)-
sigma*tp12/lambda(x)+etap(x)*r0*w*cos(w*t)/lambda(x)+r0*w*cos(w*t)*tp22
-3*(1-sigma)/2/lambda(x)*(a11*tp12+a12*tp22+a12*tp11+a22*tp12);
    dyo(3,1)=-alpha(x)*(tp12^2+tp22^2)/etap(x)-
sigma*tp22/lambda(x)-3*(1-sigma)/lambda(x)*(tp12*a12+tp22*a22);
    dyo(4,1)=-alpha(x)*(tp33^2)/etap(x)-tp33*sigma/lambda(x)-3*(1-
sigma)/lambda(x)*a33*tp33;

```

```

        dyo(5,1)=r0*w*cos(w*t)*a12+2*CI*abs(r0*w*cos(w*t))*(1.0-
3.0*a11)+chi*r0*w*cos(w*t)*a12-...
        2*chi*r0*w*cos(w*t)*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-
a22)...
        -27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-
a22))+...
        27.0*(a12^2)*(1-a11-a22))*a11*a12);
        dyo(6,1)=1.0/2.0*r0*w*cos(w*t)*a22-
1.0/2.0*r0*w*cos(w*t)*a11+chi*((1.0/2.0*r0*w*cos(w*t)*a22+...
        1.0/2.0*r0*w*cos(w*t)*a11)-
(2.0*r0*w*cos(w*t))*((27.0*a11*a22*(1-a11-a22)-...
        27.0*(a12^2)*(1-a11-a22))*(-
1.0/35.0+1.0/7.0*a11+1.0/7.0*a22))+...
        (1.0-27.0*a11*a22*(1-a11-a22)...
        +27.0*(a12^2)*(1-a11-a22))*a12^2))-
6.0*CI*abs(r0*w*cos(w*t))*a12;
        dyo(7,1)=-r0*w*cos(w*t)*a12+2*CI*abs(r0*w*cos(w*t))*(1.0-
3.0*a22)+chi*r0*w*cos(w*t)*a12-...
        2*chi*r0*w*cos(w*t)*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22)-
...
        27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-a22))+...
        27.0*(a12^2)*(1-a11-a22))*a12*a22);

end

end

```

Composite Model Predictions: Modeling the Stress Wave in SAOS Flow

```
function [export] = SAOSModeling(para,sigma,CI,freq,Gp,Gdp,modes,graph)
%This program models a small amplitude oscillatory shear (SAOS) flow
and
%graphs the resulting stress wave. Not needed in the calculation of
the
%model prediction for SAOS flow, but can be examined to verify the
accuracy
%of Kremer's method of fitting to the stress wave in SAOS flow

%From Tim Kremer: see his thesis for details on this program

x=1;
r0=0.5; %percent

c=1; %wt% CNFs, 1=2%, 2=5%, 3=10%, 4=0%

ref=10; %index for frequencys
w=freq(ref); %index for frequency

re=70.0; %aspect ratio

chi=1.0*(re^2-1)/(re^2+1); %another form of aspect ratio

%% CHECK - this is same as ext_pde
rf=1750.0; %fiber density
%% CHECK - this is same as ext_pde
rs=1000.0; %polymer density

etap=para(1,:);
lambda=para(2,:);
alpha=para(3,:);

tspan=0:1/10/w:10*pi/w;

mass=[2.0/100.0, 5.0/100.0, 10.0/100.0, 0.0];

% modes=5;
%tspan=0:0.01:shear_time;

phi=1.0*rs*mass(c)/(rf+(rs-rf)*mass(c)); %volume fraction
Ap=1.0*re^2/(3*log(sqrt(1.0*pi/phi))); %A2, shape factor

orientation=[0.333, 0.0, 0.333];

tau_final=zeros(3,modes);

while x<modes+1
    initial=[0 0 0 0 orientation];
```

```

[time, yo]=ode45(@modepolymersub,tspan,initial);
export = [' mode ' num2str(x) ' calculation done'];
disp(export)

L=length(time);

if x==1
    tau1=zeros(L,modes);
    tau12=zeros(L,modes);
    tau2=zeros(L,modes);
    tau3=zeros(L,modes);
end

tau1(:,x)=yo(:,1);
tau12(:,x)=yo(:,2);
tau2(:,x)=yo(:,3);
tau3(:,x)=yo(:,4);

r=r0*w*cos(w*time);

if graph==1
figure
subplot(1,3,1)
plot(time,tau12)
title('Stress - Individual Modes')

hold on
end

tau_final(:,x) = [tau1(L,x) tau12(L,x) tau2(L,x)]';

x=x+1;

end

total11=sum(tau1,2);
total12=sum(tau12,2);
total22=sum(tau2,2);
total33=sum(tau3,2);

a11=yo(:,5);
a12=yo(:,6);
a22=yo(:,7);
a33=1-a11-a22;

eta=total12./r;
coef=2.0*eta*phi;

if r~=0
    disp('r~=0')
    tf12=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-
a11-a22))).*(-1.0/35.0+1.0/7.0.*(a11+a22)))+(a12.^2).*(1-
27.*(a11.*a22.*(1-a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));

```

```

        tf11=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-
a11-a22))).*(3.0/7.0.*(a12)))+(a11.*a12).*(1-27*(a11.*a22.*(1-a11-
a22))+27.0.*(a12.^2).*(1-a11-a22)));
        tf22=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-
a11-a22))).*(3.0/7.0.*(a12)))+(a22.*a12).*(1-27*(a11.*a22.*(1-a11-
a22))+27.0.*(a12.^2).*(1-a11-a22)));
        tf33=(coef.*Ap.*r).*((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-
a11-a22))).*(1.0/7.0.*(a12)))+(1-a11-a22).*(1-27*(a11.*a22.*(1-
a11-a22))+27.0.*(a12.^2).*(1-a11-a22)));
else
    disp('r==0')
    tf12=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-
(a12.^2).*(1-a11-a22))).*(-1.0/35.0+1.0/7.0*(a11+a22)))+(a12.^2).*(1-
27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
    tf11=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-
(a12.^2).*(1-a11-a22))).*(3.0/7.0*(a12)))+(a11.*a12).*(1-
27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
    tf22=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-
(a12.^2).*(1-a11-a22))).*(3.0/7.0*(a12)))+(a22.*a12).*(1-
27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
    tf33=(2.0*phi.*total12.*Ap).*((27.0*(a11.*a22.*(1-a11-a22)-
(a12.^2).*(1-a11-a22))).*(1.0/7.0*(a12)))+(1-a11-a22).*(1-
27*(a11.*a22.*(1-a11-a22))+27.0*(a12.^2).*(1-a11-a22)));
end

tauc12=tf12+total12;
tauc11=tf11+total11;
tauc22=tf22+total22;
tauc33=tf33+total33;
etac=tauc12./r;
N1c=tauc11-tauc22;
N2c=tauc22-tauc33;
aijkl12=((27.0.*(a11.*a22.*(1-a11-a22)-(a12.^2).*(1-a11-a22))).*(-
1.0/35.0+1.0/7.0.*(a11+a22)))+(a12.^2).*(1-27*(a11.*a22.*(1-a11-
a22))+27.0.*(a12.^2).*(1-a11-a22)));

export=[time tauc12];

pred=Gp(ref)*0.5*sin(freq(ref)*tspan)+Gdp(ref)*0.5*cos(freq(ref)*tspan)
;

export=[time tauc12 pred'];

if graph==1
    subplot(1,3,2)
    if c==4
        plot(tspan,pred,tspan,total12)
    else
        plot(tspan,pred,tspan,tauc12)
    end
    exp=['frequency= ' num2str(w)];
    legend('Gprime Gdoubleprime','stress')
    title(exp)
    grid on
    grid minor
    hold on

```



```

subplot(1,3,3)
plot(tspan,a11,tspan,a12,tspan,a22)
legend('a11','a12','a22')
title('Orientation Evolution')
    grid on
    grid minor
end
function dyo = modepolymersub(t,y)

    tp11=y(1);
    tp12=y(2);
    tp22=y(3);
    tp33=y(4);
    a11=y(5);
    a12=y(6);
    a22=y(7);
    a33=1-a11-a22;
    dyo=zeros(7,1);

    dyo(1,1)=-alpha(x)*(tp11^2+tp12^2)/etap(x)-
sigma*tp11/lambda(x)+2*r0*w*cos(w*t)*tp12-3*(1-
sigma)*(tp11*a11+tp12*a12)/lambda(x);
    dyo(2,1)=-alpha(x)*(tp11*tp12+tp12*tp22)/etap(x)-
sigma*tp12/lambda(x)+etap(x)*r0*w*cos(w*t)/lambda(x)+r0*w*cos(w*t)*tp22
-3*(1-sigma)/2/lambda(x)*(a11*tp12+a12*tp22+a12*tp11+a22*tp12);
    dyo(3,1)=-alpha(x)*(tp12^2+tp22^2)/etap(x)-
sigma*tp22/lambda(x)-3*(1-sigma)/lambda(x)*(tp12*a12+tp22*a22);
    dyo(4,1)=-alpha(x)*(tp33^2)/etap(x)-tp33*sigma/lambda(x)-3*(1-
sigma)/lambda(x)*a33*tp33;

    dyo(5,1)=r0*w*cos(w*t)*a12+2*CI*abs(r0*w*cos(w*t))*(1.0-
3.0*a11)+chi*r0*w*cos(w*t)*a12-...
        2*chi*r0*w*cos(w*t)*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-
a22))...
        -27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-
a22))+...
        27.0*(a12^2)*(1-a11-a22))*a11*a12);
    dyo(6,1)=1.0/2.0*r0*w*cos(w*t)*a22-
1.0/2.0*r0*w*cos(w*t)*a11+chi*((1.0/2.0*r0*w*cos(w*t)*a22+...
    1.0/2.0*r0*w*cos(w*t)*a11)-
(2.0*r0*w*cos(w*t))*((27.0*a11*a22*(1-a11-a22))-...
    27.0*(a12^2)*(1-a11-a22))*(-
1.0/35.0+1.0/7.0*a11+1.0/7.0*a22))+...
    (1.0-27.0*a11*a22*(1-a11-a22))...
    +27.0*(a12^2)*(1-a11-a22))*a12^2))-
6.0*CI*abs(r0*w*cos(w*t))*a12;
    dyo(7,1)=-r0*w*cos(w*t)*a12+2*CI*abs(r0*w*cos(w*t))*(1.0-
3.0*a22)+chi*r0*w*cos(w*t)*a12-...
        2*chi*r0*w*cos(w*t)*(3.0/7.0*a12*(27.0*a11*a22*(1-a11-a22))-
...
        27.0*(a12^2)*(1-a11-a22))+(1.0-27.0*a11*a22*(1-a11-a22))+...
        27.0*(a12^2)*(1-a11-a22))*a12*a22);

end
end

```